Reconstruct your Sample

Much more informations in UK-Latin American tutorial :
https://indico.ph.ed.ac.uk/event/126/
How-tos / scripts :
https://dune-france.pages.in2p3.fr/analysis-workshop/

Laura Zambelli
Yoann Kermaïdic

DUNE-FR Analysis Workshop — April 2023

How to reconstruct your sample

From Yesterday :

```
gen_protodune_vd_cosmics.root
gen_protodune_vd_cosmics_g4_stage1.root
gen_protodune_vd_cosmics_g4_stage1_g4_stage2.root
gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim.root
```

To reconstruct the last file (with the detector simulation), type :

```
lar -c protodunevd_reco.fcl gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim.root
```

(It may take some time)

After a lot of cout, one can see what was called and how long it took :

TimeTracker printout (sec)	Min	Avg	Max	Median	RMS	nEvts
Full event	113.804	132.925	152.046	132.925	19.1211	2
source:RootInput(read)	0.000497633	0.00102209	0.00154655	0.00102209	0.000524458	2
reco:caldata:DataPrepModule	2.98958	4.8512	6.71281	4.8512	1.86162	2
reco:rns:RandomNumberSaver	2.6266e-05	0.000121289	0.000216312	0.000121289	9.5023e-05	2
reco:wclsdatavd:WireCellToolkit	86.4994	87.3345	88.1696	87.3345	0.835096	2
reco:gaushit:GausHitFinder	0.55156	0.916876	1.28219	0.916876	0.365316	2
reco:reco3d:SpacePointSolver	4.04468	10.4546	16.8646	10.4546	6.40993	2
reco:hitpdune:DisambigFromSpacePoints	0.034687	0.0959543	0.157222	0.0959543	0.0612672	2
reco:pandora:StandardPandora	2.51013	11.289	20.0678	11.289	8.77883	2
reco:pandoraTrack:LArPandoraTrackCreation	0.320205	0.787076	1.25395	0.787076	0.466871	2
reco:pandoraStdcalo:Calorimetry	0.144305	0.334414	0.524524	0.334414	0.190109	2
reco:pandoraGnocalo:GnocchiCalorimetry	0.0135242	0.0179033	0.0222823	0.0179033	0.00437906	2
[art]:TriggerResults:TriggerResultInserter	1.9183e-05	3.6416e-05	5.3649e-05	3.6416e-05	1.7233e-05	2
end_path:out1:RootOutput	5.31e-06	1.05025e-05	1.5695e-05	1.05025e-05	5.1925e-06	2
end_path:out1:RootOutput(write)	12.95	16.8072	20.6645	16.8072	3.85725	2

And a new file has been created :

```
gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root
```

Et voilà! Questions ?

Check the output file

PRO TIP

```
lar -c eventdump.fcl gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root -n 1
```

```
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE LABEL... | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE.................................................... | ..SIZE
SinglesGen.. | cosmicgenerator | ..................... | std::vector<simb::MCTruth>........................................... | .....1
SinglesGen.. | TriggerResults. | ..................... | art::TriggerResults.................................................. | .....1
G4Stage1.... | largeant....... | ..................... | sim::ParticleAncestryMap............................................. | .....-
G4Stage1.... | TriggerResults. | ..................... | art::TriggerResults.................................................. | .....1
G4Stage1.... | largeant....... | ..................... | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 596131
G4Stage1.... | largeant....... | ..................... | std::vector<simb::MCParticle>........................................ | 596131
G4Stage2.... | TriggerResults. | ..................... | art::TriggerResults.................................................. | .....1
G4Stage2.... | rns............ | ..................... | std::vector<art::RNGsnapshot>........................................ | .....1
G4Stage2.... | IonAndScint.... | ..................... | std::vector<sim::SimEnergyDeposit>................................... | .....?
Detsim...... | TriggerResults. | ..................... | art::TriggerResults.................................................. | .....1
Detsim...... | tpcrawdecoder.. | simpleSC............. | std::vector<sim::SimChannel>......................................... | .12288
Detsim...... | tpcrawdecoder.. | daq.................. | std::vector<raw::RawDigit>........................................... | .12288
Detsim...... | rns............ | ..................... | std::vector<art::RNGsnapshot>........................................ | .....0
Reco........ | TriggerResults. | ..................... | art::TriggerResults.................................................. | .....1
Reco........ | caldata........ | dataprep............. | art::Assns<raw::RawDigit,recob::Wire,void>........................... | .12288
Reco........ | pandora........ | ..................... | art::Assns<recob::PFParticle,recob::SpacePoint,void>................. | .12210
Reco........ | pandora........ | ..................... | std::vector<recob::Vertex>........................................... | ...229
Reco........ | pandora........ | ..................... | art::Assns<recob::PFParticle,recob::Slice,void>...................... | ...229
Reco........ | hitpdune....... | ..................... | art::Assns<recob::Wire,recob::Hit,void>.............................. | .15059
Reco........ | pandora........ | ..................... | std::vector<recob::Slice>............................................ | ....66
Reco........ | pandoraStdcalo. | ..................... | std::vector<anab::Calorimetry>....................................... | ...201
Reco........ | pandora........ | ..................... | art::Assns<recob::PFParticle,recob::Vertex,void>..................... | ...229
Reco........ | pandora........ | ..................... | std::vector<larpandoraobj::PFParticleMetadata>....................... | ...229
Reco........ | pandoraGnocalo. | ..................... | art::Assns<recob::Track,anab::Calorimetry,void>...................... | ...201
Reco........ | caldata........ | dataprep............. | std::vector<recob::Wire>............................................. | .12288
Reco........ | pandora........ | ..................... | art::Assns<recob::Slice,recob::Hit,void>............................. | .13828
Reco........ | pandora........ | ..................... | std::vector<recob::Cluster>.......................................... | ...467
Reco........ | pandora........ | ..................... | std::vector<recob::PFParticle>....................................... | ...229
Reco........ | pandoraTrack... | ..................... | art::Assns<recob::PFParticle,recob::Track,void>...................... | ....67
Reco........ | wclsdatavd..... | wiener............... | std::vector<recob::Wire>............................................. | .12288
Reco........ | reco3d......... | ..................... | art::Assns<recob::SpacePoint,recob::Hit,void>........................ | ..5625
Reco........ | pandora........ | ..................... | art::Assns<recob::Cluster,recob::Hit,void>........................... | .13844
Reco........ | hitpdune....... | ..................... | art::Assns<recob::Hit,recob::SpacePoint,void>........................ | ..5625
Reco........ | pandoraGnocalo. | ..................... | std::vector<anab::Calorimetry>....................................... | ...201
Reco........ | pandoraStdcalo. | ..................... | art::Assns<recob::Track,anab::Calorimetry,void>...................... | ...201
Reco........ | gaushit........ | ..................... | std::vector<recob::Hit>.............................................. | .15059
Reco........ | pandora........ | ..................... | art::Assns<recob::PFParticle,larpandoraobj::PFParticleMetadata,void>. | ...229
Reco........ | pandoraTrack... | ..................... | art::Assns<recob::Track,recob::Hit,void>............................. | .11788
Reco........ | pandoraTrack... | ..................... | std::vector<recob::Track>............................................ | ....67
Reco........ | pandoraTrack... | ..................... | art::Assns<recob::Track,recob::Hit,recob::TrackHitMeta>.............. | .11788
Reco........ | pandora........ | ..................... | art::Assns<recob::PFParticle,recob::Cluster,void>.................... | ...467
Reco........ | gaushit........ | ..................... | art::Assns<recob::Wire,recob::Hit,void>.............................. | .15059
Reco........ | reco3d......... | ..................... | std::vector<recob::PointCharge>...................................... | ..1875
Reco........ | reco3d......... | ..................... | std::vector<recob::SpacePoint>....................................... | ..1875
Reco........ | rns............ | ..................... | std::vector<art::RNGsnapshot>........................................ | .....0
Reco........ | pandora........ | ..................... | art::Assns<recob::SpacePoint,recob::Hit,void>........................ | .12210
Reco........ | hitpdune....... | ..................... | std::vector<recob::Hit>.............................................. | .15059
Reco........ | pandora........ | ..................... | std::vector<recob::SpacePoint>....................................... | .12210
Reco........ | wclsdatavd..... | gauss................ | std::vector<recob::Wire>............................................. | .12288
```

New at the reco stage

Different types of data products

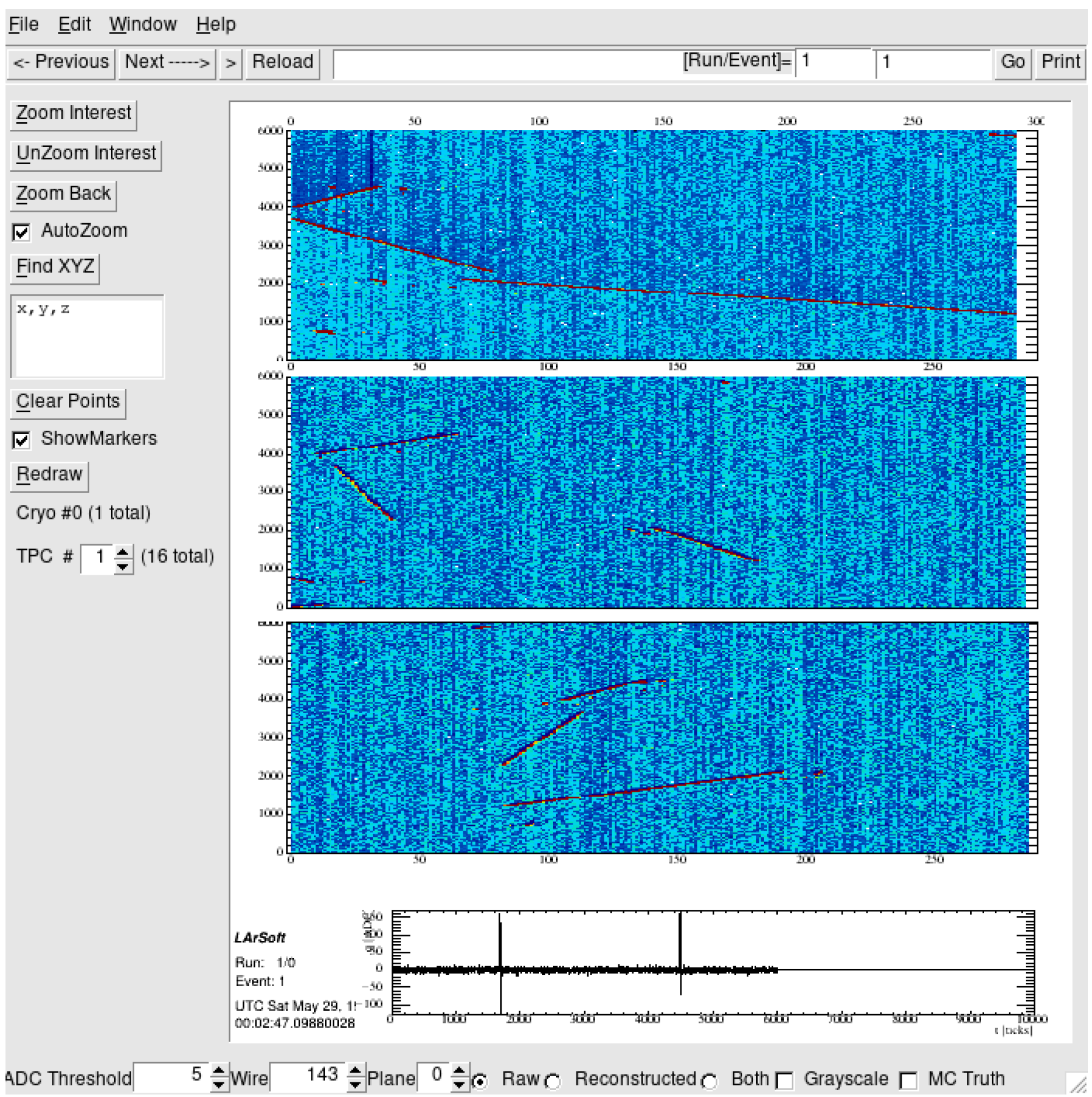
Description of data products

- raw::* - raw data
  - raw::RawDigit, raw::AuxDetDigit, raw::OpDetPulse, raw::OpDetWaveform, raw::Trigger, raw::BeamInfo, etc.
- recob::* - reconstructed information
  - recob::Wire, recob::Hit, recob::Cluster, recob::EndPoint2D, recob::Vertex, recob::PFParticle, recob::Track, recob::Shower, recob::OpHit, recob::OpFlash, etc.
- anab::* - information that is derived from reconstruction information
  - anab::Calorimetry, anab::ParticleID, anab::CosmicTag, anab::T0, etc.
- simb::* - simulation information
  - simb::MCTruth, simb::MCParticle, simb::MCFlux, etc.

Check the output file

Run the Event display for protoDUNE-VD:

```
lar -c evd_protoduneVD_driftX.fcl gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root
```



The event is split in TPC volumes

One can look at :

- raw event
- reconstructed
- both overlaid

(Latter two did not work for me)

Window > Ortho3D should gives a 3D view of the reconstructed objects

- Crashed for me

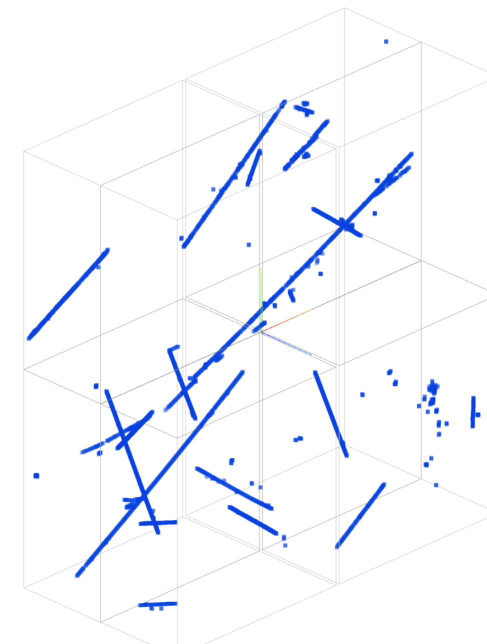
Check the output file

Check with BEE :

```
lar -c celltree_protodunevd.fcl gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root
dunesw/fcl/protodunevd/view/upload-to-bee.sh bee/bee_upload.zip
```

```
Django Auth: get csrftoken ... lAOEuamni6tx5Fb9rdZnGdYF4U75A55b
uploading file bee/bee_upload.zip ...
https://www.phy.bnl.gov/twister/bee/set/42c26001-2e42-4b6c-8ddd-45278e080a04/event/list/
```

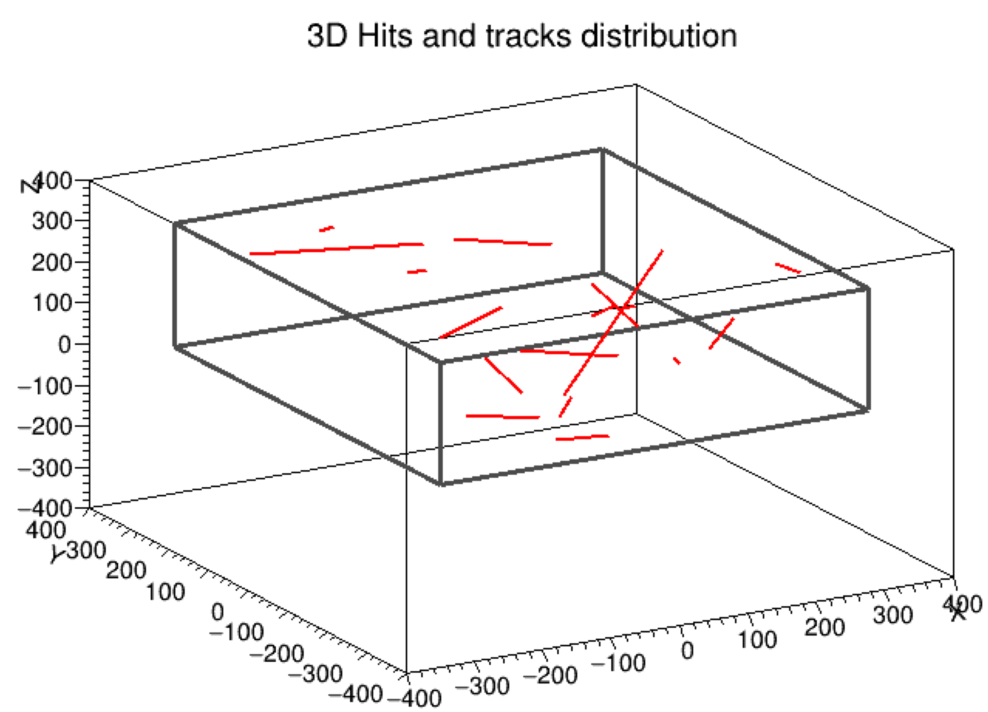
↳ go to the link, where you'll have an interactive 3D view of your reconstructed event in your browser



Check with CheckHitsAndTracks Module of Thibaut :

```
lar -c pdvd_hits_and_tracks.fcl gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root
```

A file `hits_and_tracks_results.root` is created, and amount many useful parameters contains the 3D event display of the reconstructed tracks



Check the output file

With a script:

Minimally working example :

Track.h class

```
R__ADD_INCLUDE_PATH("lardataobj/RecoBase/Track.h")
R__ADD_INCLUDE_PATH("gallery/Event.h")

using namespace art;

void my_script(){
  string filename = "gen_protodune_vd_cosmics_g4_stage1_g4_stage2_detsim_reco.root";
  vector<string> filenames(1, filename);

  string trktag = "pandoraTrack";
  InputTag track_tag(trktag);

  size_t eventnb = 0;

  for (gallery::Event ev(filenames); !ev.atEnd(); ev.next()) {
    cout << " EVENT " << eventnb << endl;

    auto const& tracklist = *ev.getValidHandle<vector<recob::Track>>(track_tag);
    size_t ntracks = tracklist.size();
    cout << " has " << ntracks << " tracks reconstructed " << endl;

    for (size_t itrk=0; itrk<ntracks; ++itrk) {
      const recob::Track t = tracklist[itrk];
      cout << itrk << " Start point : (" << t.Start().X() << ", " << t.Start().Y() << ", " << t.Start().Z() << ") "<< endl;
    }

    eventnb++;

  }

}
```

```
 EVENT 0
 has 67 tracks reconstructed
0 Start point : (301.934, 166.312, 142.351)
1 Start point : (-184.479, 56.8635, 298.894)
2 Start point : (-187.653, -64.6903, 35.4855)
3 Start point : (49.7706, 335.184, 145.453)
```

(...)

WARNING :This is NOT AN ANALYSIS SCRIPT !
You should make a module for that !

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

Fcl tutorial

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

-> Not re-inventing the wheel :
Include generic/detector specific fcl files
and tune the parameters

PRO TIP: To check the full parameters of a fcl file:

```
fhicl-dump protodunevd_reco.fcl
```

(NB : the output can be very long!)

In protodunevd_reco.fcl :

```
#include "services_refactored_pdune.fcl"
#include "caldata_dune.fcl"
#include "wirecell_dune.fcl"
#include "hitfindermodules_dune.fcl"
#include "SpacePointSolver_dune.fcl"
#include "cluster_dune.fcl"
#include "trackfindermodules_dune.fcl"
#include "pandoramodules_dune.fcl"
#include "calorimetry_vdcb.fcl"
#include "calorimetry_pdune.fcl"
#include "calibration_dune.fcl"
#include "featurelabelingmodules.fcl"
#include "particleid.fcl"
#include "mctrutht0matching.fcl"
#include "t0reco.fcl"
```

(...)

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

[Fcl tutorial]

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

process_name: -> Unique name in your analysis chain of your process

In our example, each stage has its own process_name:
Generation : SingleGen
G4 : G4stage1, G4stage2
detsim : Detsim
Reconstruction : reco

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

[Fcl tutorial]

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

→ Services are tools commonly used
E.g: geometry, physical properties, file management

In protodunevd_reco.fcl :

```
services:
{
  TFileService: { fileName: "reco_protoDUNE_hist.root" }
  TimeTracker:       @local::dune_time_tracker
  MemoryTracker:     @local::dune_memory_tracker
  RandomNumberGenerator: {} #ART native random number generator
  message:              @local::dune_message_services_prod
  FileCatalogMetadata:  @local::art_file_catalog_mc
                        @table::protodunevd_reco_services
  #ChannelStatusService: @local::pdsp_channel_status
  IFDH: {}
  DetectorPropertiesService:  @local::protodunevd_detproperties
}



services.BackTrackerService.BackTracker.G4ModuleLabel: "largeant"
services.BackTrackerService.BackTracker.SimChannelModuleLabel: "tpcrawdecoder:simpleSC"

services.RawDigitPrepService.ToolNames: @local::pdsim_dataprep_tools_wirecell
```

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

[Fcl tutorial]

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

What is your input type, default name, how many events to process (-1 means everything)

In protodunevd_reco.fcl :

```
source:
{
  module_type: RootInput
  maxEvents: 100
  saveMemoryObjectThreshold: 0
  fileNames: ["detsim_single_protoDUNE.root"]
}
```

Other cases examples :

For MC generation :

```
module_type: EmptyEvent
```

For Top-CB Data Reconstruction :

```
module_type: VDColdboxTDERawInput
```

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

[Fcl tutorial]

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

Define and configure the modules to be called on your events
(i.e. where the magic happens) -> two types of actions : mo

```
physics:{
  producers: {}
  filters : {}
  analyzers: {}

  stream1 : []
  trigger_paths : []
  end_paths : []
}
```

- producers ← Generates new informations to be added the products
- filters ← Removes informations
- analyzers ← Transforms informations (plots, histograms, ...)
- stream1 ← Where to store the information
- trigger_paths ← List of actions with impact on the products
- end_paths ← List of actions with no impact on the products

The fcl files

fcl stands for FHICL = Fermilab Hierarchical Configuration Language
-> Configure and run LArSoft modules
-> Written in a JSON-like language
-> Set key values (i.e. no hard-coded parameters in LArSoft)
-> Can change fcl parameters without recompilation

Fcl tutorial

A fcl should contains these 6 elements :

```
#include
process_name:
services: {}
source : {}
physics : {}
outputs : {}
```

The output file(s) to be produced : type, name, what to keep/discard ('drop')

In protodunevd_reco.fcl :

```
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "%ifb_reco.root"
   dataTier:    "full-reconstructed"
   outputCommands: [ "keep *", "drop *_reco3d_noreg_*", "drop *_reco3d_pre_*" ]
   compressionLevel: 1 #zlib argument (0-9)
   fastCloning: true
 }
}
```

The fcl file - services

A service is a generic algorithm/tool

```
services:
{
  TFileService: { fileName: "reco_protoDUNE_hist.root" }
  TimeTracker:       @local::dune_time_tracker
  MemoryTracker:     @local::dune_memory_tracker
  RandomNumberGenerator: {} #ART native random number generator
  message:              @local::dune_message_services_prod
  FileCatalogMetadata:  @local::art_file_catalog_mc
                        @table::protodunevd_reco_services
  #ChannelStatusService: @local::pdsp_channel_status
  IFDH: {}
  DetectorPropertiesService:  @local::protodunevd_detproperties
}



services.BackTrackerService.BackTracker.G4ModuleLabel: "largeant"
services.BackTrackerService.BackTracker.SimChannelModuleLabel: "tpcrawdecoder:simpleSC"

services.RawDigitPrepService.ToolNames: @local::pdsim_dataprep_tools_wirecell
```

E.g `pdsim_dataprep_tools_wirecell` service is defined here :

```
# ProtoDUNE sim dataprep converted to ke scale.
pdsim_dataprep_tools_calib: [
  "digitReader",                # Read RawDigit
  "adcSampleFiller",            # Subtract pedestal, trivial calibration
  "adcScaleAdcToKe"             # Scale samples to nominal ADC counts
]

# ProtoDUNE sim dataprep at ke scale with tail removal.
pdsim_dataprep_tools_oldtail: [
  @sequence::pdsim_dataprep_tools_calib,
  "adcCorrectUndershootKe"      # correct undershoot with old tool
]

# ProtoDUNE sim dataprep at ke scale with tail removal.
pdsim_dataprep_tools_tail: [
  @sequence::pdsim_dataprep_tools_calib,
  "pdspTailPedRemovalZKe"       # correct tails
]

# Drop ROIs, scale back to ADC and zero bad/noisy channels for wirecell processing.
pdsim_dataprep_tools_wirecell: [
  @sequence::pdsim_dataprep_tools_tail,
  "adcKeepAllSignalFinder",     # Keep all signal (no ROIs)
  "adcScaleKeToAdc",            # Scale samples to nominal ADC counts
  "pdsp_RemoveBadChannels"      # Set bad channels to 0 ADC
```

NB : here we are simulating & reconstructing simulation with no correlated noise injected

↳ The dataprep stage here is fairly simple, while it can be more complicated for real data

https://github.com/DUNE/dunesw/blob/develop/fcl/vdcoldbox/reco/crpcb_top_process.fcl

Producers of the our reco fcl

The chain of producers is defined in `trigger_paths`

e.g. : Generic `producer_adcprep_notool` described here :

```
# DataPrepModule is the producer that calls RawDigitPrepService

# This configuration does not make use of the decoding tool.
# Input is raw::RawDigit vector DigitLabel in the event data store.
# Use with MC. Can be used with data if the decoder module is called first.
producer_adcprep_notool: {
  module_type:   "DataPrepModule"
  LogLevel:                     1
  DecoderTool:                 ""       # Non-blank reads digits with named tool.
  DigitLabel:               "daq"       # Label for input digits if decoder tool is not used
  TimeStampName:               ""       # Non-blank writes RDTimeStamps if decoder tool is used
  OutputDigitName:             ""       # Non-blank writes digits if decoder tool is used
  WireName:                    ""
  DoAssns:                   true
  DoGroups:                  true
  ChannelRanges:               []
  BeamEventLabel:              ""
  IntermediateStates:          []
  OnlineChannelMapTool:  pd_onlineChannelMapByFemb
}
```

Generic parameters can be changed in your fcl, e.g.:

```
physics.producers.caldata.DigitLabel: "tpcrawdecoder:daq"
physics.producers.caldata.WireName: "dataprep"
physics.producers.caldata.LogLevel: 3
```

In protodunevd_reco.fcl :

```
physics: {

  filters: {
    trigfilter:  @local::pdsp_trigfilter_all
    nhitsfilter: @local::standard_numberofhitsfilter
  }

  producers: {
    rns:              { module_type: "RandomNumberSaver" }
    caldata:          @local::producer_adcprep_notool
    wclsdatavd:       @local::protodunevd_nfsp
    #wclsdatanfsp:     @local::dune_vd_coldbox_nfsp
    gaushit:          @local::dunevdfd_gaushitfinder
    reco3d:           @local::protodunespmc_spacepointsolver

    # actual disambiguation
    hitpdune:         @local::pdune_disambigfromsp
    emtrkmichelid:    @local::protodune_emtrkmichelid

    pandora:          @local::dunefdvd_pandora_cosmic
    pandoraTrack:     @local::dunefdvd_pandoraTrackCreation
    pandoraShower:    @local::dunefdvd_pandoraModularShowerCreation

    #pandora:          @local::protodune_pandora
    #pandoraTrack:     @local::dune_pandoraTrackCreation
    #pandoraShower:    @local::dune_pandoraShowerCreation

    pandoraWriter:    @local::dune_pandorawriter
    pandoraStdcalo:   @local::pdune_vd_standard_calodata
    pandoraGnocalo:   @local::pdune_vd_gnocchi_calodata
  }

reco: [ caldata,
        rns,
        wclsdatavd,
        gaushit,
        reco3d,
        hitpdune,
        pandora,
        pandoraTrack,
        # pandoraShower,
        pandoraStdcalo,
        pandoraGnocalo #,
        ]

  stream1:  [ out1 ]

  trigger_paths: [reco]

  end_paths:     [stream1]
}
```

Producers of the our reco fcl

In our case the producers called are :

1. caldata: prepare the waveforms
2. rns: random number module (for reproducibility)
3. wclsdatavd: Wirecell call (mostly deconvolution)
4. gaushit: search & fit the hits with gaussian within waveform
5. reco3d: build (x,y,z) out of matching hits in multiple views
6. hitpdune: Disambiguates the hits

7.8. pandora, Track, (Shower): builds tracks/shower out of found hits

9.10. pandoraStdcalo, pandoraGnocalo : computes the dQ/ds from the found tracks

In protodunevd_reco.fcl :

```
physics: {

  filters: {
    trigfilter:  @local::pdsp_trigfilter_all
    nhitsfilter: @local::standard_numberofhitsfilter
  }

  producers: {
    rns:             { module_type: "RandomNumberSaver" }
    caldata:             @local::producer_adcprep_notool
    wclsdatavd:          @local::protodunevd_nfsp
    #wclsdatanfsp:        @local::dune_vd_coldbox_nfsp
    gaushit:             @local::dunevdfd_gaushitfinder
    reco3d:              @local::protodunespmc_spacepointsolver

    # actual disambiguation
    hitpdune:            @local::pdune_disambigfromsp
    emtrkmichelid:       @local::protodune_emtrkmichelid

    pandora:             @local::dunefdvd_pandora_cosmic
    pandoraTrack:        @local::dunefdvd_pandoraTrackCreation
    pandoraShower:       @local::dunefdvd_pandoraModularShowerCreation

    #pandora:             @local::protodune_pandora
    #pandoraTrack:        @local::dune_pandoraTrackCreation
    #pandoraShower:       @local::dune_pandoraShowerCreation

    pandoraWriter:       @local::dune_pandorawriter
    pandoraStdcalo:      @local::pdune_vd_standard_calodata
    pandoraGnocalo:      @local::pdune_vd_gnocchi_calodata
  }

reco: [ caldata,
        rns,
        wclsdatavd,
        gaushit,
        reco3d,
        hitpdune,
        pandora,
        pandoraTrack,
        # pandoraShower,
        pandoraStdcalo,
        pandoraGnocalo #,
        ]

  stream1:  [ out1 ]

  trigger_paths: [reco]

  end_paths:     [stream1]
}
```

NB : Some producers are defined but not called in trigger_paths

What happened - data preparation

Binary decoder — *Data only* -> Handled by services

- -

Channel Mapping

Baseline subtraction -> Histogram each waveform & fit a gaussian

Noise Filtering

-CNR
-Microphonics
-Sticky Code
- ...

- CNR (Coherent Noise Removal):
  Removes noise seen at the same time by a group of channel
  -> Compute & subtract the median noise in group
  Group = same electronic card and same view
  2 CNRs : normal and weighted
  -> Weighted by channel RMS
- Microphonic :
  Removes slow pedestal fluctuations in space and time
  -> Sinusoid pattern smoothed (LOWLESS method)
- Other waveform/noise corrections possible:
  - Sticky code : Digitizer sticks to certain ADC value
  - Tail removal : Long-term response of the electronics

MicroBoone paper on noise

ProtoDUNE-SP paper

PRO TIP: To keep the waveforms, change this command in your fcl file:

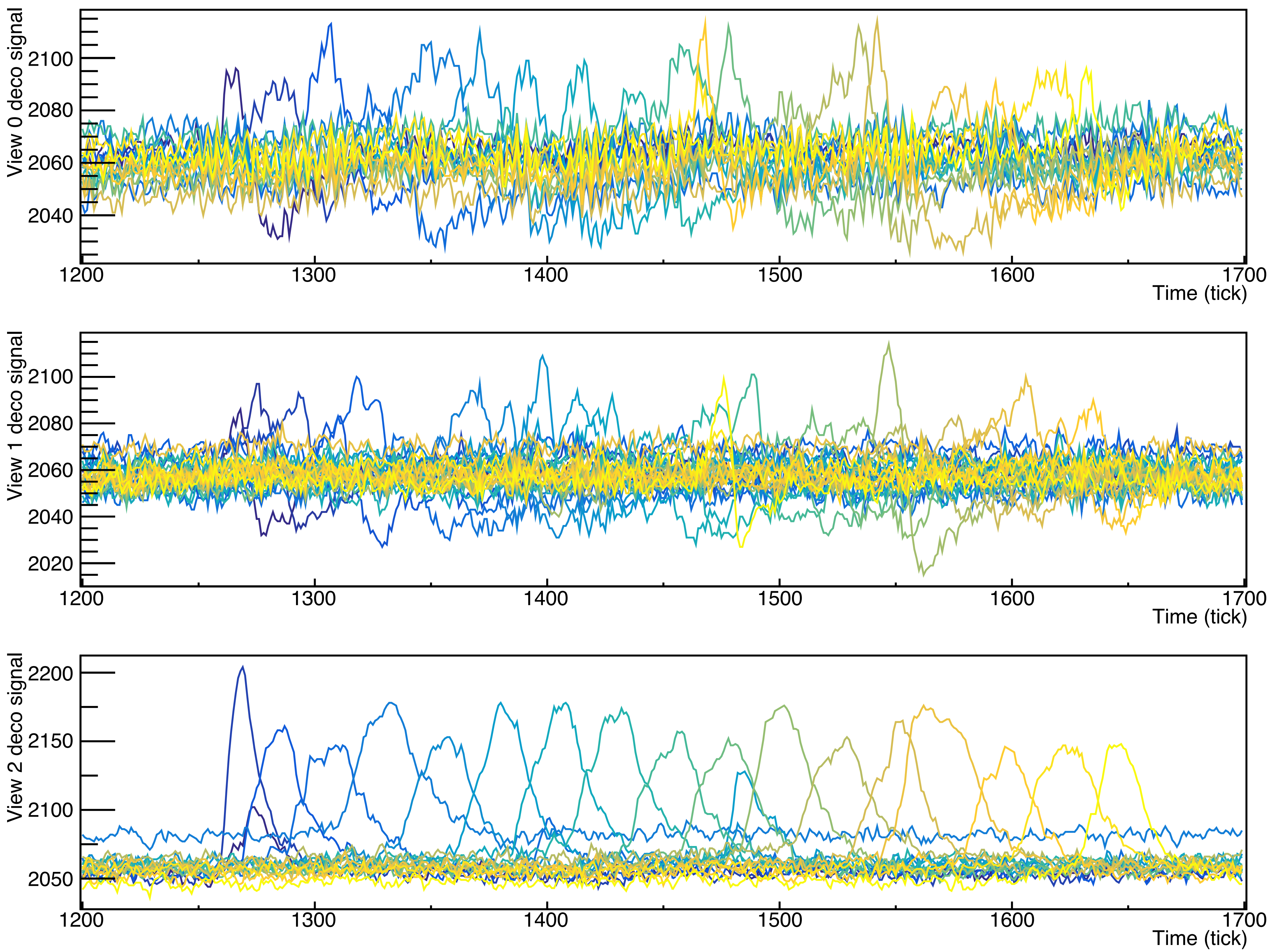
```
#outputCommands: [ "keep *", "drop raw::RawDigit*_*_*_*" ]
outputCommands: [ "keep *"]
```

What happened - Raw waveform

PRO TIP Use Yoann's script ! `PlotDecoPulses.C`

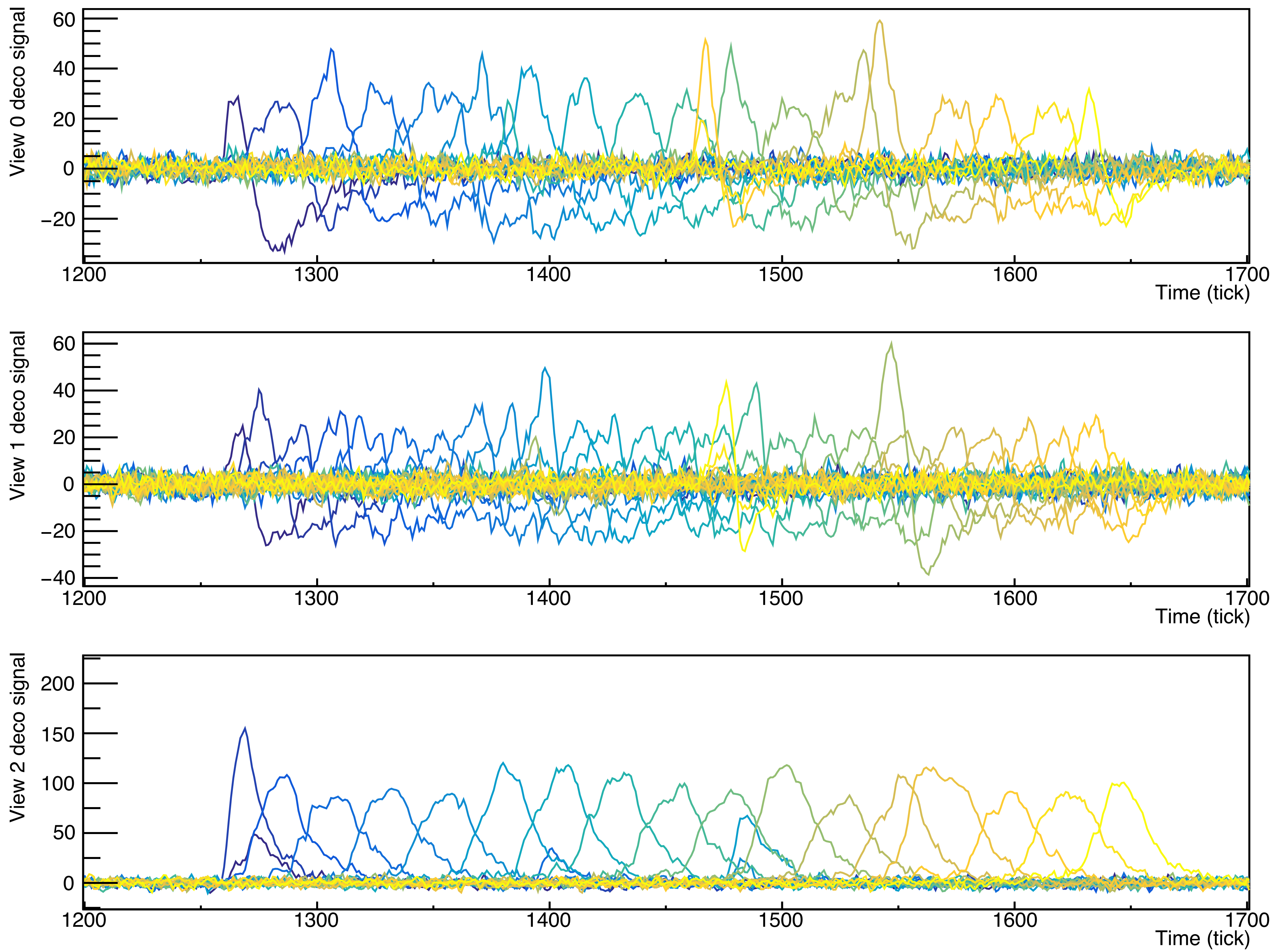
Example for ColdBox CRP1 data :

-> Raw waveform



What happened - After CNR

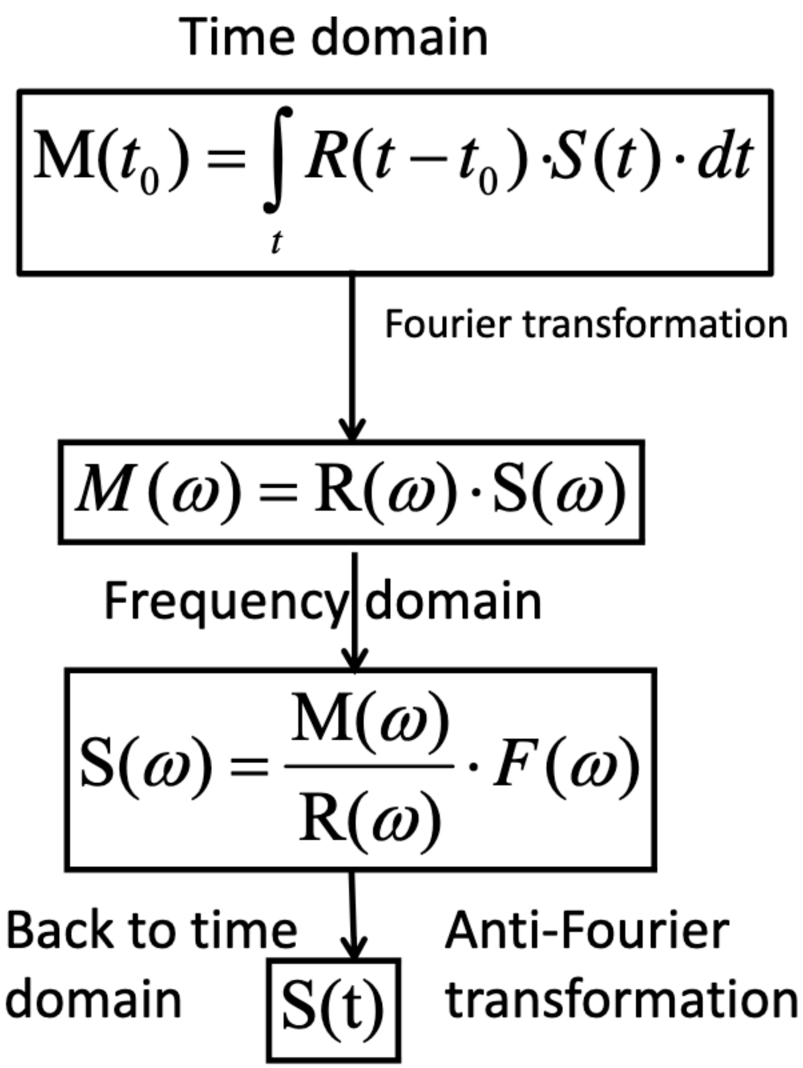
Example for ColdBox CRP1 data :
-> pedestal subtracted and CNR applied



What happened - Deconvolution

-> MicroBoone paper on signal processing part I and part II

Take-away message : This part is complex and would require a dedicated workshop on its own!



This part is done in Wirecell :

The measured output M is a convolution of the true signal S with a function R
-> R contains the electronic response function and the field response

To retrieve the true signal, an FFT is performed.

F(ω) is a filter function (Gauss or Wiener) to mitigate the high frequency noise after deconvolution

The output of wirecell is *zero padded* :
- -> The noise region are set to 0
- -> the waveforms are the same length as the input

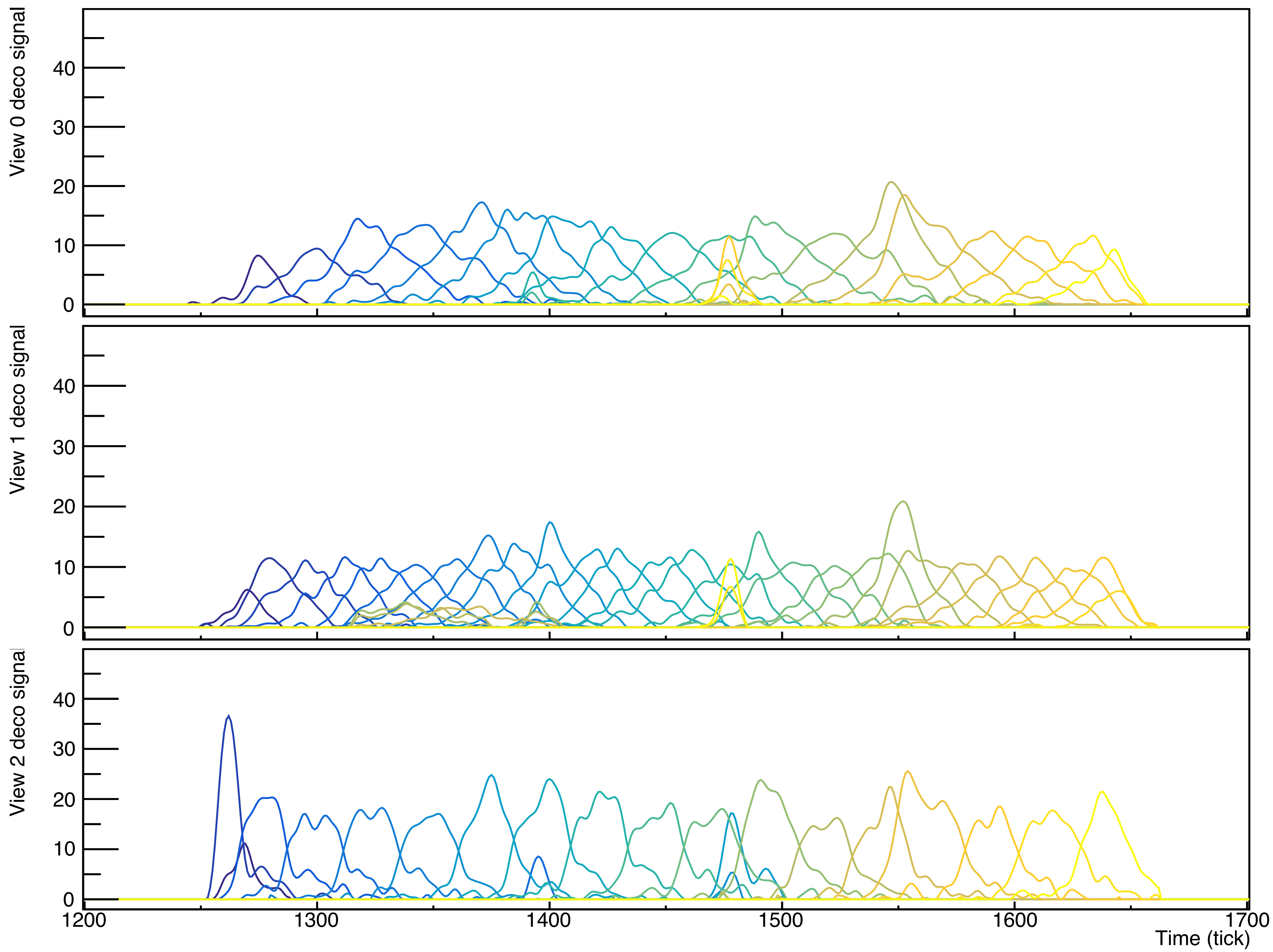
Wirecell have its own ROI finder:
- -> Hardcoded thresholds at the moment
- -> Could need some fine VD-tuning

Wirecell deconvolution still to be improved :
- -> PCB- Weighting Field
- -> Top and Bottom electronics

What happened - Deconvolution

Example for ColdBox CRP1 data :

-> Waveform deconvolved : no more bipolar signal ; zeros where no signal



What happened - Hit Finder

Gaushit

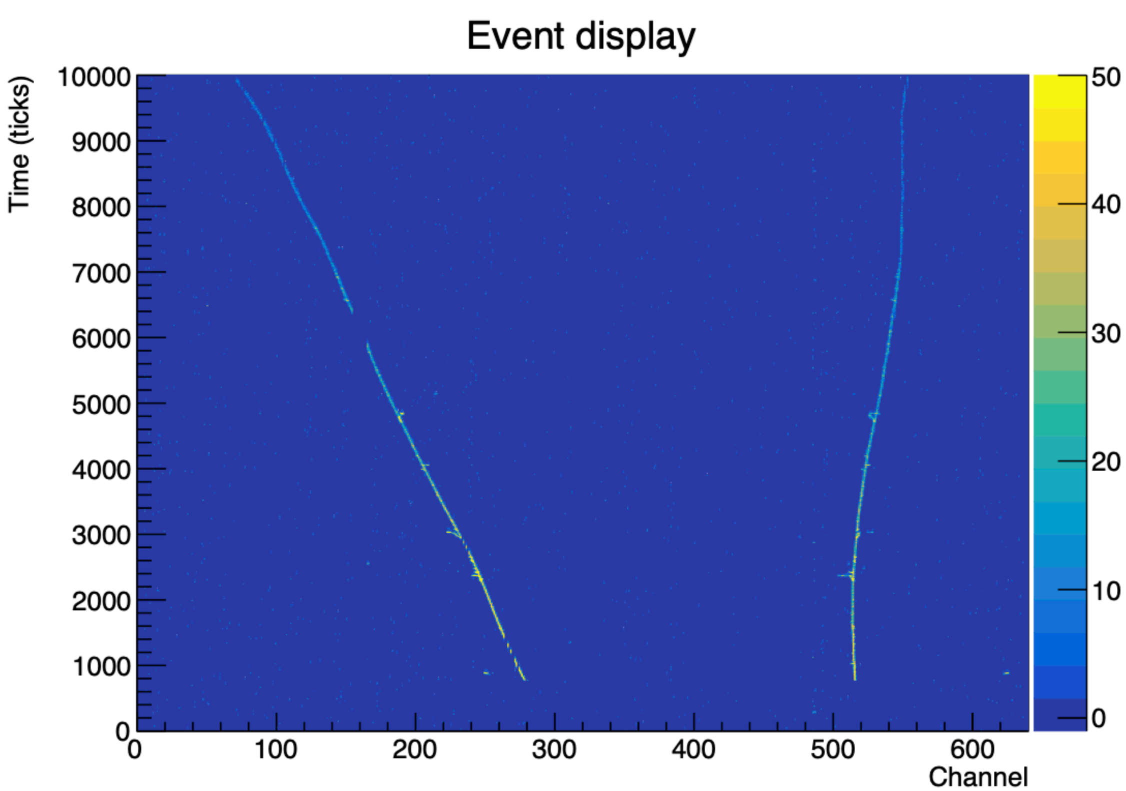
- searches for signal ≥ Threshold × Channel noise RMS

Thresholds are defined in the fcl files:

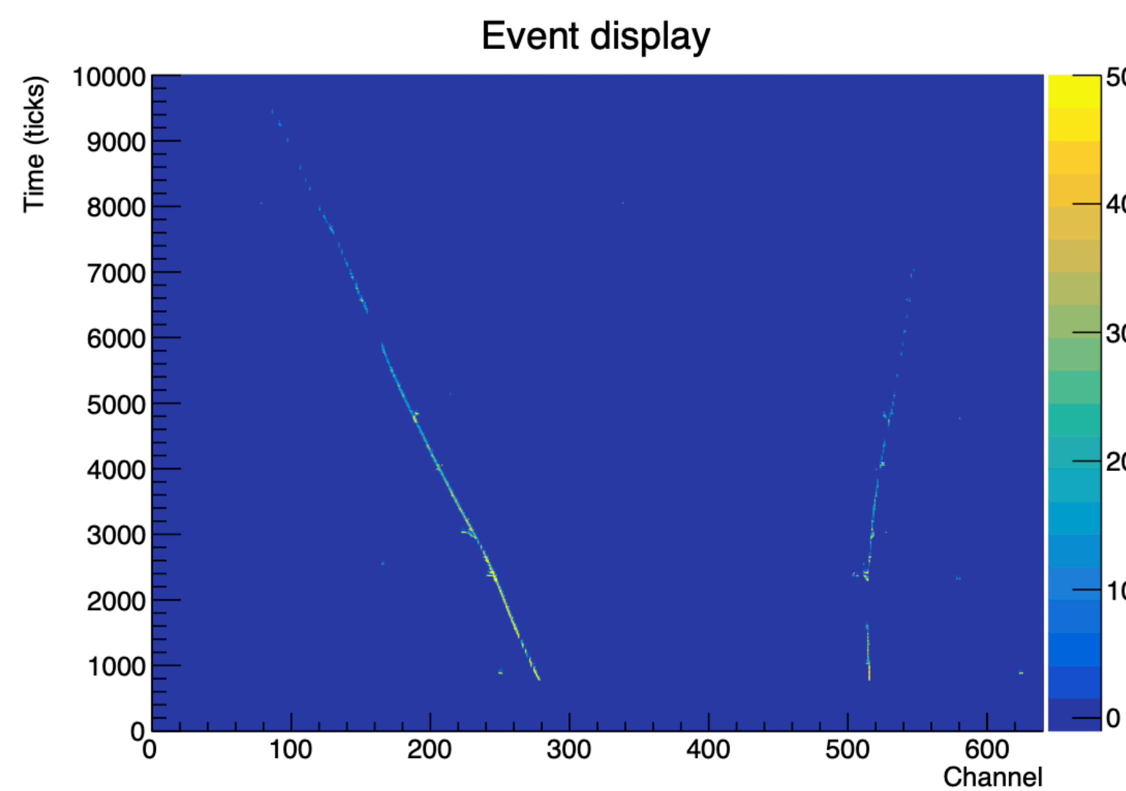
```
physics.producers.gaushit.HitFinderToolVec.CandidateHitsPlane0.RoiThreshold: 5.0
physics.producers.gaushit.HitFinderToolVec.CandidateHitsPlane1.RoiThreshold: 5.0
physics.producers.gaushit.HitFinderToolVec.CandidateHitsPlane2.RoiThreshold: 5.0
```

Example (6m drift data)

Low ROI threshold



High ROI threshold



What happened - Hit Finder

Gaushit

- searches for signal ≥ Threshold × Channel noise RMS
- Fits them with gaussian (one or multiple)

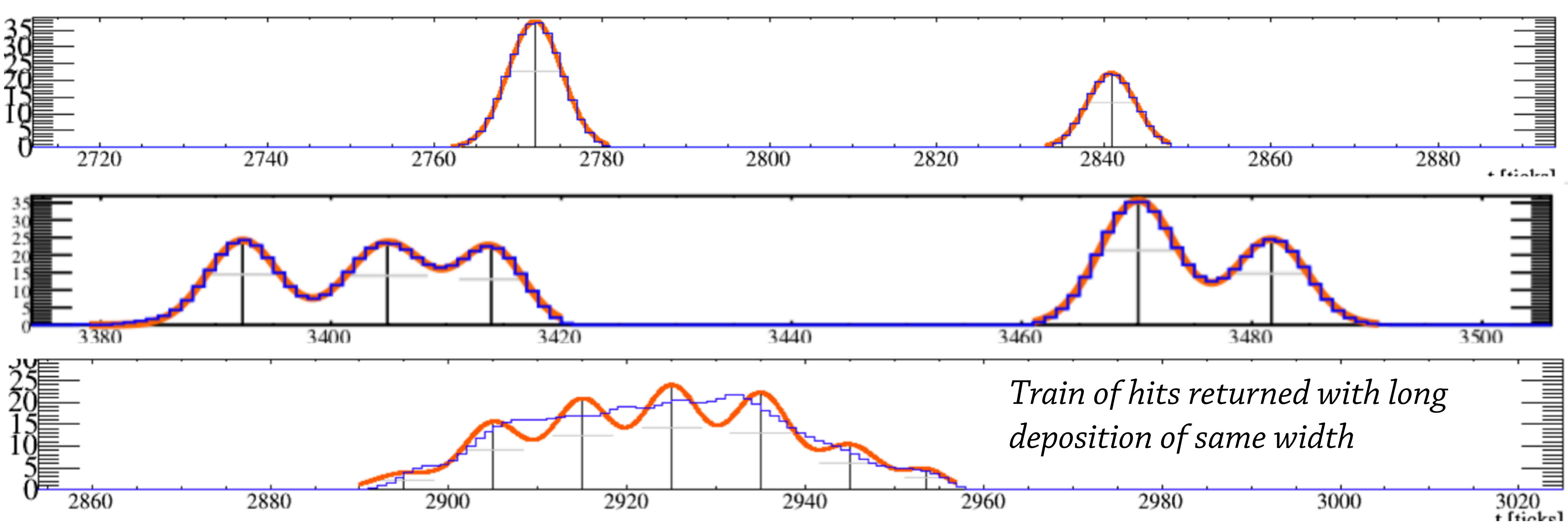
`dunefd_gaushitfinder` defined <u>here</u>:

```
dunevdfd_gaushitfinder.InitWidth: [6.0, 6.0, 6.0]
dunevdfd_gaushitfinder.AreaNorms: [13.25, 13.25, 13.25]
dunevdfd_gaushitfinder.MaxMultiHit: 4
dunevdfd_gaushitfinder.Chi2NDF: 50
dunevdfd_gaushitfinder.LongMaxHits: [ 25, 25, 25 ]
dunevdfd_gaushitfinder.LongPulseWidth: [ 10, 10, 10 ]
dunevdfd_gaushitfinder.PeakFitter:      @local::peakfitter_mrqdt
dunevdfd_gaushitfinder.CalDataModuleLabel: "tpcrawdecoder:gauss"
```

The peaking time and area of the pulse(s) are then stored

-> The fit initialization parameters are tunable in the fcl files

Same Gaus fit examples :



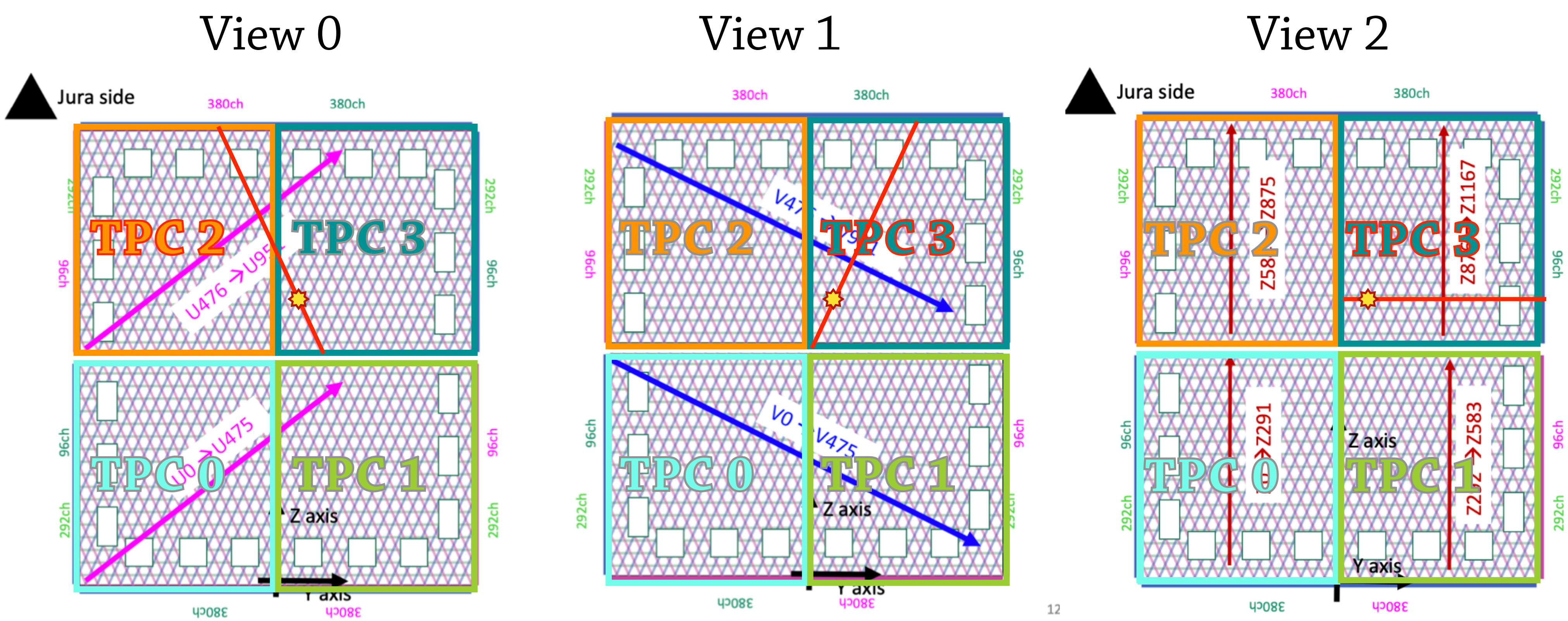
Side Note on Disambiguation

For technical reasons, a CRP is treated as four volumes in LArSoft

-> The separation follow the collection mapping

-> In the induction views some channels are split into two wires in two volumes

Can be problematic : a deposition can be seen in different volumes



→A ***disambiguation*** process is needed to reattribute hits to their correct volumes

-> Done by `hitpdune` and/or `SpacePointSolver`

Searches for depositions at similar times in the three planes and tries to match them

- Assign Hits to the correct volume
- `SpacePointSolver` computes the corresponding (x,y,z)

EXERCICE : try to reconstruct the event without disambiguation !

What happened - Reconstruction

Goes from hit collection (in {channel, time} or {x,y,z}) to set of hierarchical showers, tracks

-> The reconstruction is done by Pandora - which can be seen as a black box for most of us

« Pandora is a multi algorithm approach »

Three main reconstruction steps :

1. Clustering : Assemble hits in the same plane together based on proximity/topology
2. Matching: clusters across planes merged together based on granularity, geometry, topology and training from 2D-patter recognition
3. Hierarchy: Identify and organize the tracks/showers into a history (example : K->π->μ->e)

For ProtoDUNE data, two reconstruction chains:

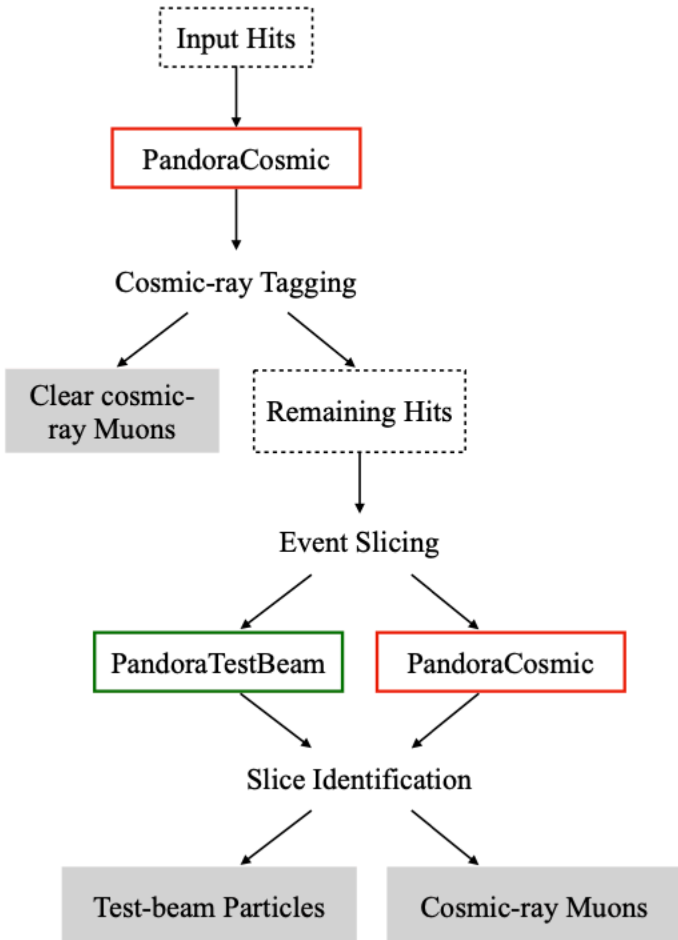
-> Beam : Particles from known position in space and time, usually with many generations

-> Cosmic : Muon-like tracks from above

[For FD a neutrino event reconstruction chain is called]

What happened - Reconstruction

ProtoDUNE-SP Pandora paper



Procedure for ProtoDUNE-SP:
Clear cosmic are identified with a set of criteria :

-> very vertical, t_0 not in time with the beam, track enters from top of the volume, part of the track is out of time if $t_0 = t_{beam}$, …

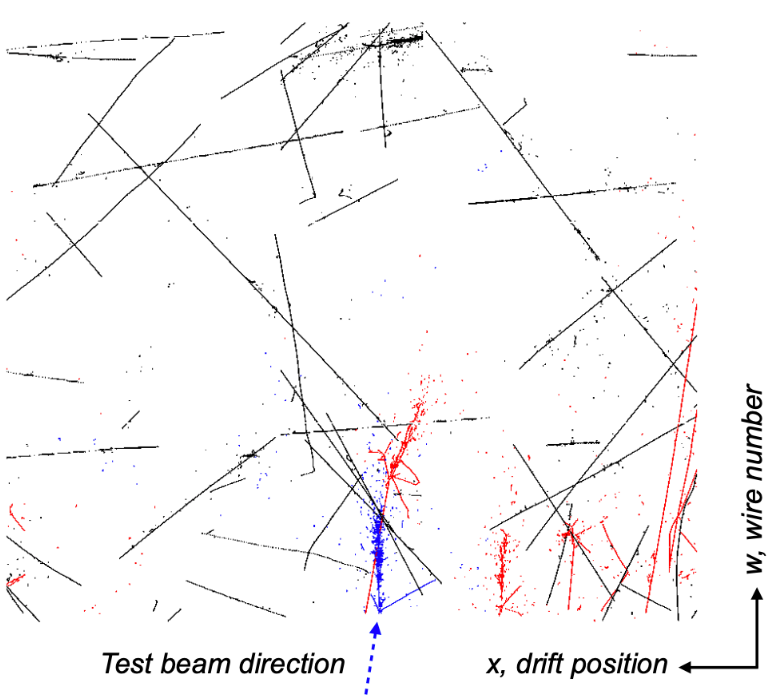
-> Clear cosmic are identified and « removed »

All unclear objects (« slice ») are analyzed further under the « Beam » or « Cosmic hypothesis

-> A BDT helps to distinguish one hypothesis from an other (one outcome per slice)

Example of BDT inputs:

- Distance from the object endpoints to the beam pipe entrance
- Direction of the object wrt to the beam line
- Distance to the closest detector boundary, …



Cosmic
Beam (π^+)
Beam Halo

Pandora needs to be adapted for Module-0 data:
- Beam location and direction
- e^- drift direction/ CR topology
- Profit from Ghost tracks