Support for 2D channel-space reconstruction

Gianluca Petrillo, Saba Sehrish, Erica Snider

University of Rochester/Fermilab

LArSoft Coordinators Meeting, July 14th , 2015

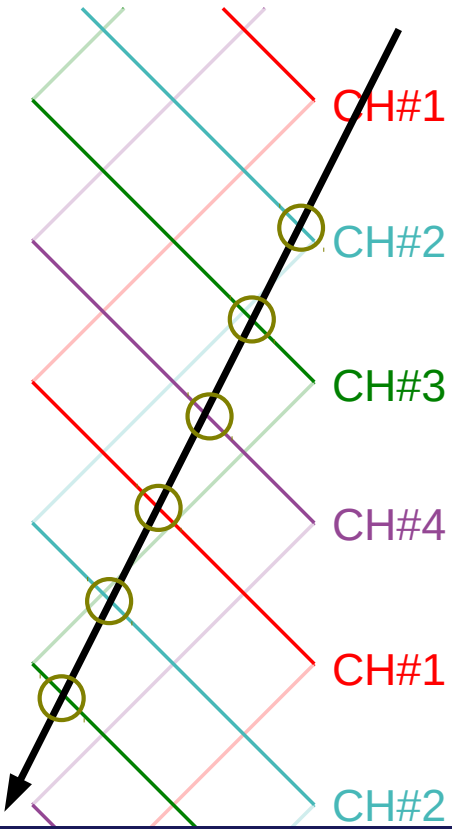
Strange case of TPCs and wrapped wires

Definitions:

wire a wire segment lying on a plane of a TPC

channel readout channel connected to one or more wires

In the simplest TPC assemblies (“ArgoNeuT-like”) there is a one-to-one correspondence...

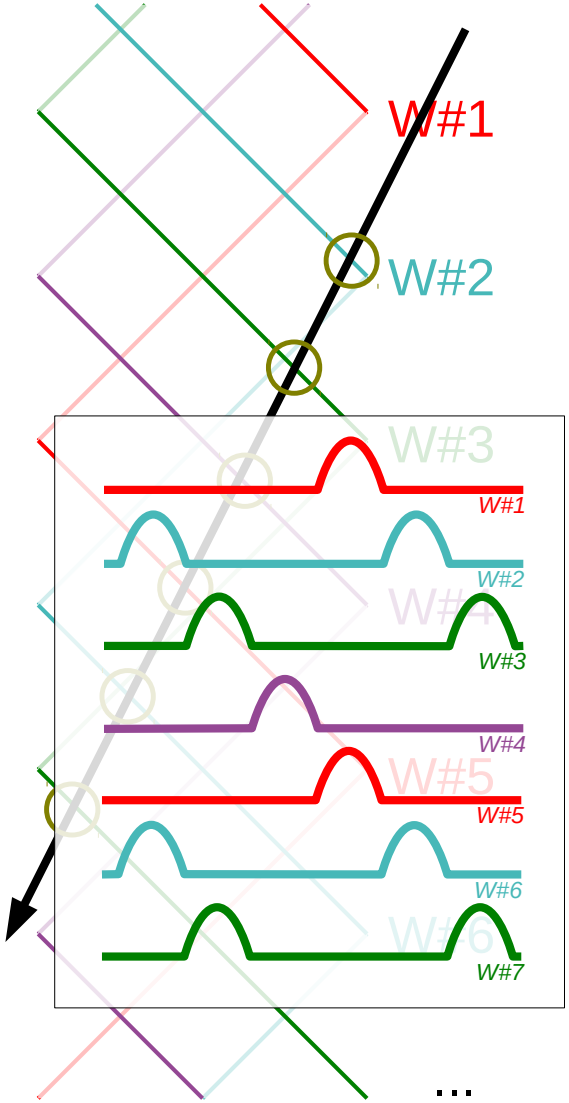


On the left, we represent a more complicate ones (“DUNE-like”), where channels cover multiple wires in different TPCs.

The descending wires that go leftward belong to the TPC in the backyard but they see the signal of this one.

In this cartoon, there is no activity from the backyard TPC. That’s frequently not the case.

Simple reconstruction in geometry space

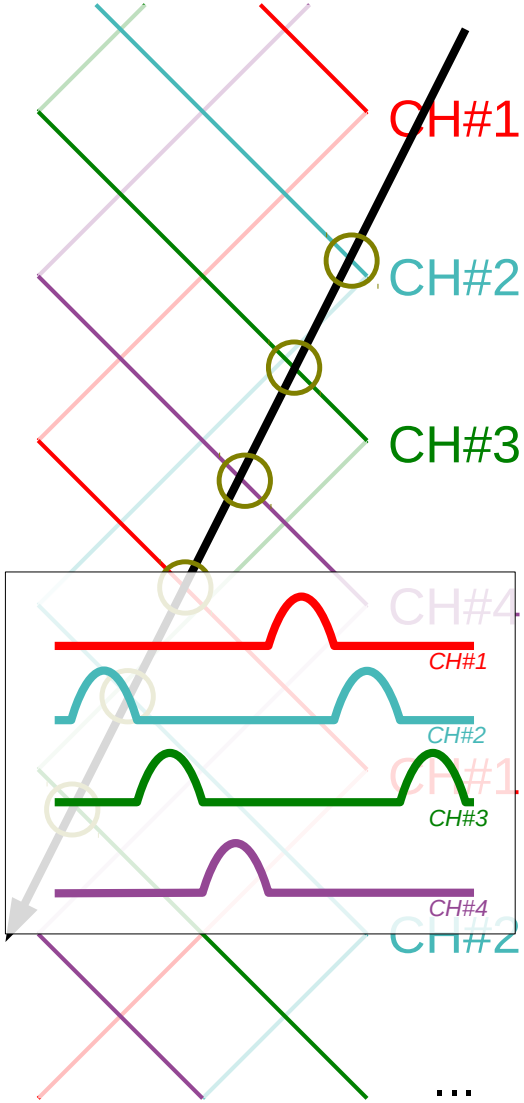


Interpreting the hits in wire space:

- wires in this TPC share the same signal
- the track is split in segments (because of the periodicity of the channels in the physical space)
- there are ghost segments (because wires reflect the signal from elsewhere)

Here we are taking a track that travels away from the wire plane. It's more intuitive without adding the complication of TDC time vs. real time...

Simple reconstruction in readout space



Interpreting the hits in channel space:

- channels pick signal from the same particle multiple times (because each channel appears in multiple places)
- the track is split in segments (because of the periodicity of the channels in the physical space)
- these segments are not pinned to the physical space

Action items

So far, we have only reconstruction in wire space.
We proposed to add structures to facilitate the reconstruction in channel space.

The following guidelines were approved[1]:

- add identification of readout entities
- extend geometry interface to map readout and geometry entities
- make 2D objects aware of their location in readout space too
- follow the new policy for these 2D reconstructed objects:
  - readout location is always defined and valid
  - geometry location is defined if not ambiguous, invalid otherwise

[1] LArSoft Architecture meetings on June 17th and June 24th , 2015.

New readout IDs

The following IDs were introduced:

group of TPCs `TPCsetID` contains a cryostat ID and a TPC set index

readout plane `ROPID` contains a TPC set ID plus a readout plane index

channels the existing `raw::ChannelID_t` (a plain integer) is retained

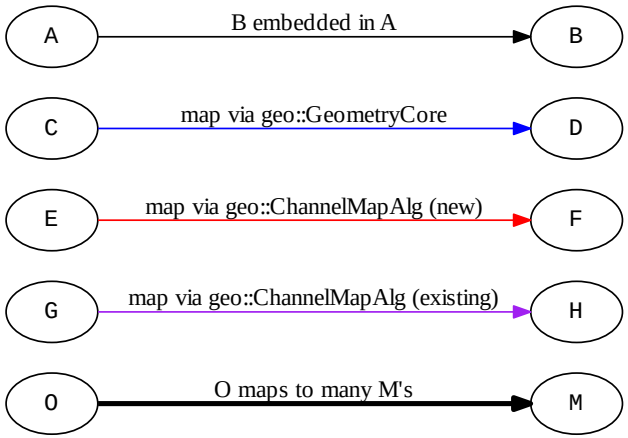
cryostat `CryostatID` is an alias of `geo::GeometryID`

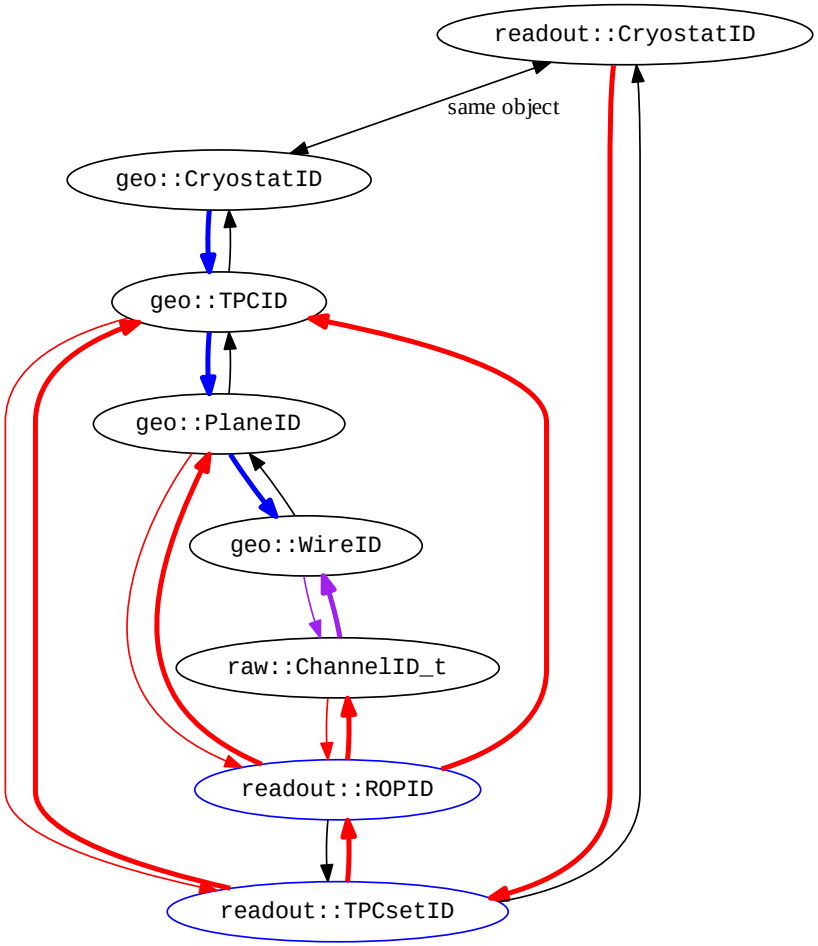
Each of these IDs uniquely represents a readout entity.
These IDs have similar functionalities as the geometry counterpart.

The new ID classes live in `readout` namespace and are declared in `larcore/SimpleTypesAndConstants/readout_types.h`.

New mapping

Mapping is defined by `geo::ChannelMapAlg` interface, accessed via `geo::GeometryCore`:





Each of these IDs uniquely represents a readout entity.
These IDs have similar functionalities as the geometry counterpart.

New data and policies

Data products have been updated:

`recob::Cluster` readout plane added: `ROPID()`

`recob::EndPoint2D` readout plane added: `ROPID()`

`recob::Hit` channel information is still in `Channel()`

More important is the new policy to fill them:

new data members should always be filled and valid

`recob::Hit` wire ID will be invalid unless the hit location is unambiguous

`recob::Cluster` wire plane ID will be invalid unless the cluster location is unambiguous

`recob::EndPoint2D` no policy change (by necessity)

Breaking changes: data format

The data product format is backward-compatible

`recob::Hit` existing hits are read correctly; but *they might not follow the new policy*

`recob::Cluster` existing clusters are read correctly; but *they do not follow the new policy* since the ROP ID is marked invalid

`recob::EndPoint2D` existing 2D vertices are read correctly; but *they do not follow the new policy* since the ROP ID is marked invalid

Breaking changes: assumptions

Breakage: `recob::Hit`

- if an existing algorithm relies on `recob::Hit::WireID()` to learn e.g. the view of the hit, it will fail on new data products when the hit is "ambiguous" *[wrapped wire detectors only]*
- if an updated algorithm relies on `recob::Hit::WireID()` validity to learn whether the hit is ambiguous, it will fail on old data products where the ID is always valid *[all detectors]*

Breakage: `recob::Cluster` *(and, similarly, `recob::EndPoint2D`)*

- if an existing algorithm relies on `recob::Cluster::Plane()` to learn e.g. the view of the cluster, it will fail on new data products when the cluster is "ambiguous" *[wrapped wire detectors only]*
- if an updated algorithm relies on `recob::Cluster::ROP()` to learn e.g. the readout plane of the hit, it will fail on old data products where that information is invalid *[all detectors]*

Status and plans

Matter of thought:

- `recob::EndPoint2D` stores the coordinate as wire ID: it might be better to add a channel ID rather than a readout plane ID

What is ongoing now:

- draft code published in branch `feature/gp_WrappedGeometry`
- expect DUNE to lead the update of their code
- some testing with MicroBooNE data is very important

What's in the pipeline (copied from two weeks ago!):

- factorization of `DetectorProperties` and `LArProperties`
- implementation of the new channel filtering model
- ... and much more (heeelp!)

Backup

Breakage fixes: `recob::Hit`

If an existing algorithm relies on `recob::Hit::WireID()` to learn e.g. the view of the hit, it will fail on new data products when the hit is "ambiguous"

- always check wire ID validity!
- if ID is invalid:
  - to get the view use `Hit::View()`
  - to get information about the location, map `Hit::Channel()` to it

If a new algorithm relies on `recob::Hit::WireID()` validity to learn whether the hit is ambiguous, it will fail on old data products where the ID is always valid

There is no way for the updated code to know whether the hit is a unambiguous "new" one or a possibly ambiguous "old" one. Assume the former and be aware that old data might require old software.

Breakage fixes: `recob::Cluster`

If an existing algorithm relies on `recob::Cluster::Plane()` to learn e.g. the view of the cluster, it will fail on new data products when the cluster is "ambiguous"

- always check plane ID validity!
- if ID is invalid:
  - to get the view use `Cluster::View()`
  - to get information about its location, map `Cluster::ROPID()` to it

If a new algorithm relies on `recob::Cluster::ROP()` to learn e.g. the readout plane of the hit, it will fail on old data products where that information is invalid

In the old data, `Cluster::Plane()` is always valid and holds some good information. If `Cluster::ROP()` is not valid, then the data *is* old, and your code can use `Cluster::Plane()` as a fall back; add a comment explaining that.