Access to tracking data products via “proxies”

An introduction.

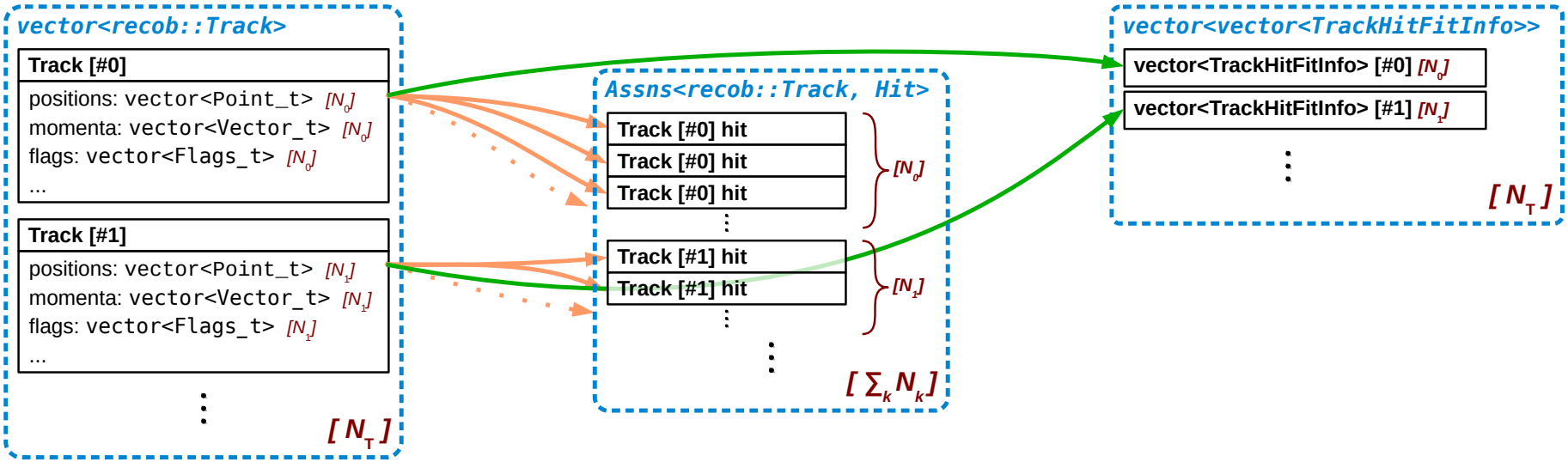
Gianluca Petrillo

Fermi National Accelerator Laboratory

LArSoft Coordination Meeting, October 10th, 2017

Tracking data products

Track reconstruction and fitting produces several data products, e.g.:



`recob::Track` trajectory points (flagged), length, ...

`recob::Hit` *(associated)* one hit per trajectory point

`recob::TrackFitHitInfo` *(parallel)* point-by-point fit information

Track proxy

A track proxy object (denoted with `proxy::Tracks`) can navigate through these connections:

1. create a proxy object "merging" the information needed; e.g.:

```
#include "lardata/RecoBaseProxy/Track.h"
// ...
auto const tracks = proxy::getCollection<proxy::Tracks>
  (event, "pandoranu", proxy::withFitHitInfo());
```

(hit information is always merged in the proxy)

2. use its interface to get access to the merged information

```
auto const track = tracks[0];                  // random access to tracks
double const length = track.track().Length(); // access recob::Track
auto const firstPoint = track.point(0);    // random access to point:
art::Ptr<recob::Hit> const& firstHit = firstPoint.hitPtr();
double const firstHitMeas = firstPoint.fitInfoPtr()->hitMeas();
```

(access to `recob::Track` members can happen also directly via `operator->()`: `double const length = track->Length();`)

Track proxy magics

The collection proxy called `proxy::Tracks` has specialised

- access to associated hits *(implicitly merged)*
- access to trajectory points *(integrated)*
- access to fit information *(requesting `proxy::withFitHitInfo()`)*

Merging other information is possible, but only the common proxy interface is offered. For example, assuming starting vertices are associated to tracks:

```
auto const tracks = proxy::getCollection<proxy::Tracks>(
  event, art::InputTag("pandoranu"),
  proxy::withFitHitInfo(), proxy::withAssociated<recob::Vertex>()
  );
// ...
auto const track = tracks[0];
auto const vertices = track.get<recob::Vertex>();
recob::Vertex const* vertex
  = vertices.empty()? nullptr: *(vertices.first());
```

(a different input tag may be specified for each merged component)

Type of supported data product structures

Currently, proxies support only a single data product as main collection (e.g. a `recob::Track` collection).

Also, proxies support merging of only two models of data products:

in-order associations where associated objects (e.g. hits) are stored so that the ones associated to the first main object (e.g. track) come first, then all the ones associated to the second one, and so forth

parallel data collections whose the first object (e.g. vector of track hit fit information) is connected to the first main object (e.g. track), the second object to the second main object, and so forth, with no holes allowed

LArSoft recommends to use one of these models as often as possible.

How to use the track proxy

Documentation is available in Doxygen format:

- each proxy is documented in a Doxygen "module"
- the module for tracking proxy points to `proxy::Tracks`
  - start from the `proxy::Tracks` page, with usage examples and documentation of single track and single track point proxy
- the generic proxy interface is also documented via examples
- *using a proxy object as function argument needs templates*; quirks like this are also documented in Doxygen

Need help? ask away!
Documentation written by the authors is seldom satisfactory to the users.

Summary

- we have a new “collection proxy” infrastructure in place
  - extensible (new data products require no special support)
  - customisable (`proxy::Tracks` is in fact a customisation)
  - usability will be tested on you!
- documentation is in Doxygen (this talk is no documentation!)
- comments and suggestions are welcome
- the first proxy being introduced in LArSoft is about tracking
  - no LArSoft code has been updated to use it yet
- more may follow (e.g. space point with charge is a possible idea)