Next generation tracking data products

A proposal

Giuseppe Cerati *Gianluca Petrillo* Erica Snider

Fermi National Accelerator Laboratory

LArSoft Coordinators' Meeting, January 17th , 2017

The “old” tracking data product

All the tracking information is currently stored in `recob::Track`

- it is still “version 1”, the first attempt
- it contains:
  - 3D points along the trajectory
  - 3D directions (one per point)
  - momentum modulus
  - covariance matrix for both track ends (format not specified)
  - a numeric ID
  - dQ/dx per plane and per point
  - also, it is associated to hits

The declaration is in `lardataobj/RecoBase/Track.h`.

What's wrong with it

This class is the first attempt to represent a track:

- it mixes different concepts:
  - detected information ("pattern")
  - fitted information (trajectory, momenta, uncertainties)
  - calorimetry (dQ/dx)
- it is produced by pattern recognition algorithms
  (which have no idea about momentum)
- it is produced by track fitting algorithms
  (which have little to no idea about calorimetry)
- it is *not* produced by calorimetry algorithms
  (which use the specific object `anab::Calorimetry`)
- it still *lacks* some information
  - order of hits along the track
  - fit quality for fitters
  - fit residuals
  - ...

Goals of the new revision

We want to redesign the data products involved with tracking:

- disentangle different concepts into separate classes
  - $\Rightarrow$ in particular: separate pattern recognition and track fitting
    - and leave calorimetry to specialised algorithms
- give algorithms the place where to put all they produce...
- ... without forcing them to make up missing information
- define exactly what analisers can expect in each data product

Some requirements for the solution and the transition:

- user code will need to be adapted
- utilities should be available to minimise the impact
- agreement on which part of old data needs to be readable after transition

Today we are starting the definition of these requirements.

The proposal: new classes

The proposal includes:

`recob::Trajectory` representing the pattern recognition result: a set of hits, and an identified path
- momentum information is *optional*
- a nominal direction is defined, but no guarantee is offered

`recob::Track` representing the result of fit on the pattern recognition result
- no room for calorimetry information here

The envisioned usage pattern:

- simple 3D pattern recognition algorithms can still produce just `recob::Trajectory`
- algorithms without their fitter are recommended to run the "standard" Kalman fitter to produce `recob::Track`
- algorithms which rely only on trajectory information will use `recob::Trajectory` objects as input

The proposal: `recob::Trajectory`

The content of the new `recob::Trajectory`:

- a trajectory made of 3D points
- momentum or direction information
  - user will ask whether momentum is available, or just directions

New requirements

1. at least two points in the trajectory
2. one *trajectory point* per associated `recob::Hit`
3. hits are associated in trajectory point order

Interface change respect to "old" `recob::Track`:

- changed name!
- low to moderate: most methods still present as legacy
- some information moved out (see next slide)
- but using a better representation for vectors (ROOT GenVector)

The proposal: `recob::Track`

The content of the new `recob::Track`:

- an updated trajectory (as `recob::Trajectory`)
- particle ID hypothesis the algorithm used
- quality of fit result
- quality flags? (to be discussed)
- covariance matrices

Same requirements as in `recob::Trajectory`!

Dropped from the "old" `recob::Track`:

- track identification number (ID)
- calorimetry information

Additional data may be associated to the track:

- e.g., residuals, covariance matrix point by point

Implementation plan

We propose a implementation plan in three stages:

1. this stage should be in time for next MicroBooNE production
   - it should include additional information on fit quality
   - January 2017? (it means less than two weeks)
   - changes are mostly “hidden” behind the interface
   - very low impact on users/developers
2. the complete implementation of the separation
   - separation of `recob::Track` and `recob::Trajectory` interfaces
   - including code update and basic tools supporting the creation of the new objects
   - with basic utilities to navigate the new objects
   - large impact on users/developers
3. full support
   - including fully featured utilities to navigate the new objects
   - low impact on users/developers

Implementation: first stage

- introduce the class `recob::Trajectory` in its "final" form
- have an *intermediate* version of `recob::Track` which:
  - *temporary* encapsulates dQ/dx information
  - exposes `recob::Trajectory` interface as its official one
  - *temporary* replicates most of the "old" `recob::Track` legacy interface (as deprecated)
  - can be loaded out of a old `recob::Track` branch
    (but then it won't fulfil the requirements!)

$\Rightarrow$ part of the old interface disappears

- mandatory changes in the code are expected to be formal only
- legacy interface is *slower* than the official one

$\Rightarrow$ track producers will need to be updated in order to fulfil the new track requirements

Summary and discussion

- to enact stage 1 within the correct time frame, we need to act quick
  (and the work is *a lot*)
- we need an agreement on:
  - vision
  - content
  - implementation stages

Discussion is open.

Backup

Fitted track: trajectory points and hits

The requirement inherited from the trajectory spells out:

"one *trajectory point* per associated `recob::Hit`"

There is an alternative proposal to be considered *for fitted tracks*:

- one trajectory point *per trajectory intersection to wires*
- points not associated with hits will need a null hit pointer