



Writing helpers for recob::Tracks

Giuseppe Cerati

LArSoft Coordination meeting

Oct. 10, 2017

Introduction

- New `recob::Track` and related data objects discussed in January
 - `recob::TrackTrajectory` is the pattern recognition output (positions and directions)
 - `recob::Track` is a fitted `recob::TrackTrajectory` (with covariance and chi2)
- Products went into LArSoft v06_23_00

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

G. Petrillo (FNAL) New tracking data products LArSoft coord, Jan 17th, 2017 5 / 12

TrackTrajectory object

- The only change with respect to the original plan so far is the introduction of a third object: `recob::TrackTrajectory`
- It inherits from `recob::Trajectory` and is stored in `recob::Track`
- It extends the simple `Trajectory` by adding a set of flags (bitmask)
- There is one bitmask for each hit/point pair
 - 1-1 correspondance (hit/point) becomes 1-1-1 (hit/point/flagset)
- Each flag can be in one of three states:
 1. the algorithm decided the flag condition is true (flag `_set_`)
 2. the algorithm decided the flag condition is false (flag `_unset_`)
 3. the algorithm declined to decide a state to the flag (flag `_undefined_`)

3 2017/01/31 cerati@fnal.gov - LArSoft Coordination meeting 

- But old constructors not deprecated and producers not updated
 - Everybody still producing `recob::Tracks` (both as pattern reco and fit product)
- This presentation addresses helper classes to facilitate the transition to the new schema.

Conventions used in recob::Track

- The following conventions are assumed in the design of the new objects:
 - 1-1 correspondence between TrajectoryPoint (i.e. position and direction) and Hits
 - Hits are associated (via Assn) in TrajectoryPoint order
 - There is a set of flags (bitmask) for each hit/point, indicating how hits and points are used in the TrackTrajectory or Track
 - TrackTrajectory objects has to be created with at least two TrajectoryPoints
- In most cases these are not hardly enforced in the code, but we provide helpers to facilitate the production of objects compliant with the convention

Overview of the helper code (bottom-up)

- Namespace for track making code, trkmkr::
- Helper code for book keeping of outputs and conventions:
 - TrackTrajectoryCreationBookKeeper, class for TrackTrajectory conventions
 - OptionalOutputs and OptionalPointElement, structs for book keeping of optional Track-related output objects
 - TrackCreationBookKeeper, class for Track book keeping
- art Tools for track fitting:
 - TrackMaker, base abstract class
 - KalmanFilterFitTrackMaker, concrete tool to fit tracks with TrackKalmanFitter
 - people are welcome to implement their favorite track fitting tool!
- Track producers using TrackMaker tools:
 - TrackProducerFromTrackTrajectory, TrackProducerFromTrack, TrackProducerFromPFParticle

Book Keeping code

- `TrackTrajectoryCreationBookKeeper`
 - Helper class to aid the creation of a `recob::TrackTrajectory`, keeping data vectors (`Point_t`, `Vector_t`, `PointFlags_t`, `Hit`) in sync. Elements of those vectors are added sequentially using the `addPoint` functions. Once all points have been added a call to the function `finalizeTrackTrajectory`, builds the track moving the content of the vectors.
- `OptionalOutputs` and `OptionalPointElement`
 - This struct holds the optional outputs of track making and hides their details to the actual track making tools. In this way, adding a new optional output will affect only those tools that produce such new output. `OptionalPointElement` holds the elements of `OptionalOutputs` that are added for each point (i.e. each hit).
- `TrackCreationBookKeeper`
 - Helper class to aid the creation of a `recob::Track`, keeping output data in sync (same data as `TrackTrajectoryCreationBookKeeper`, plus `OptionalOutputs`). It internally stores and uses a `TrackTrajectoryCreationBookKeeper` object. Elements of data vectors are added sequentially using the `addPoint` functions. Once all points have been added a call to the function `finalizeTrack`, builds the `Track` moving the content of the vectors.

art Tools for track fitting

- TrackMaker
 - Base abstract class for tools used to fit tracks.
 - The purely virtual function *makeTrack* is responsible for building the Tracks. Its arguments are the const inputs (`TrackTrajectory`, `TrajectoryPointFlags`, `trackID`) and the non-const outputs (mandatory: *Track* and *Hits*; optional outputs stored in `OptionalOutputs`).
 - Other (virtual) versions of *makeTrack* accept different inputs and call the purely virtual one
 - In case other products are needed from the event (e.g. associations to the input), they can be retrieved overriding the *initEvent* function.
 - The tool is not meant to put collections in the event.
- KalmanFilterFitTrackMaker
 - Concrete implementation of a tool to fit tracks with `TrackKalmanFitter`; inherits from abstract class `TrackMaker`. It prepares the input needed by the fitter (momentum, `particleId`, `direction`), calls the fitter, and returns a track with outputs filled.

Track producers

- Producers differ in the input object and are similar in the other features:
 - The mandatory outputs are: the resulting `recob::Track` collection, the associated hits, and the association between the input object and the output `Track`.
 - Optional outputs are `recob::TrackFitHitInfo` and `recob::SpacePoint` collections
 - plus the `Assn` of `SpacePoints` to `Hits`.
 - An option is provided to create `SpacePoints` from the `TrajectoryPoints` in the `Track`.
Note: in this context `SpacePoints` should not be used as the functionality of 3D points associated to a `Track (Trajectory)` is covered by `TrajectoryPoints`.
 - The fit is performed by an user-defined tool, which must inherit from `TrackMaker`
- Further details:
 - `TrackProducerFromPFParticle` takes as input an existing `recob::PFParticle` collection and refits the associated `Tracks`; it can make track fits of associated `Showers` as well, but this is experimental: do at your own risk
 - `TrackProducerFromTrack`: in this case `Assn` to the input object cannot be created and users should rely on the track *ID*

Survey of current recob::Track producers

- Made a quick survey of track producers currently being used, at least in uboone: Pandora, PMA, KHit, KalmanTrack
- KalmanTrack is compliant with the new requirements
- KHit fits tracks so it should definitely produce recob::Tracks; it still uses the old (soon-to-be-deprecated) constructor and needs to be updated, possibly using TrackCreationBookKeeper
- Pandora and PMA are more complicated.
 - Not standalone track producers
 - They do not produce fully fitted tracks
 - Possibly require breaking changes, see next slide

Possible options for introducing the use of TrackTrajectories

- There are 3 options on how deal with Pandora and PMA.
 - Option 1 is the preferred by the larsoft team, let us know your feedback.
1. In principle they should produce TrackTrajectory instead of Track, and the corresponding fitted Track should be produced as a next step by a dedicated producer module (if needed).
 - Breaking change: there won't be anymore the same data product with the same label (including all Assns!).
 2. Alternatively, we could call the track fitter from inside the Pandora/PMA module and produce both TrackTrajectory and Track objects.
 - Warning: the same data product with the same label may still be present, but with different meaning/properties; also adding the fitter to pandora/pma is a rather intrusive change.
 3. As 2., but overwriting fitted TrajectoryPoints with the ones from the input TrackTrajectory. In this way the code currently being used would keep the same meaning
 - Warning: not clean, covariance matrices won't correspond to start/end TrajectoryPoints

Conclusions

- Helper code to produce `recob::TrackTrajectories` and `recob::Tracks` is available since LArSoft v06_49_00
- Ready to move forward and deprecate old constructors?