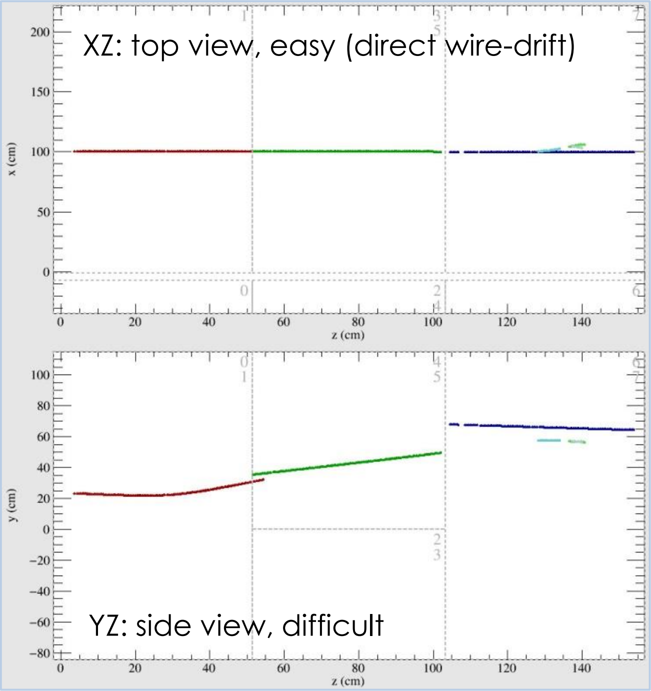
Processing tracks:

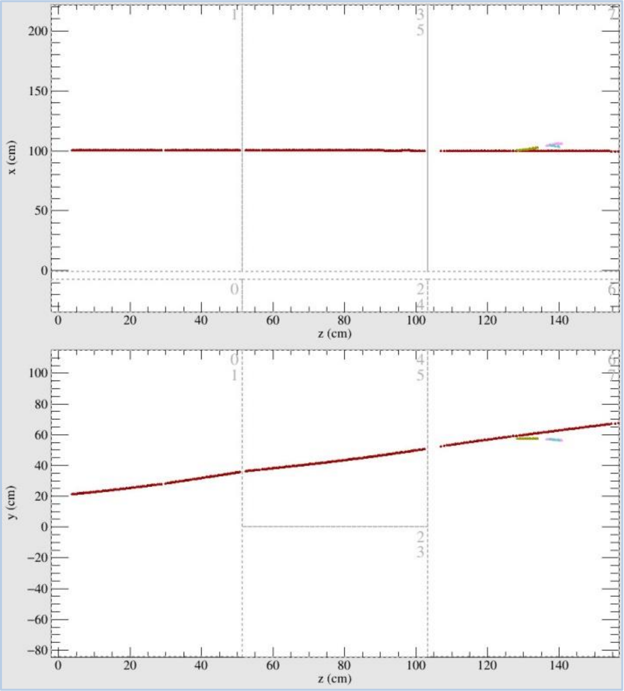
- stitching updates
- data products for tracks

R.Sulej

Wire plane parallel tracks (this was already mentioned)



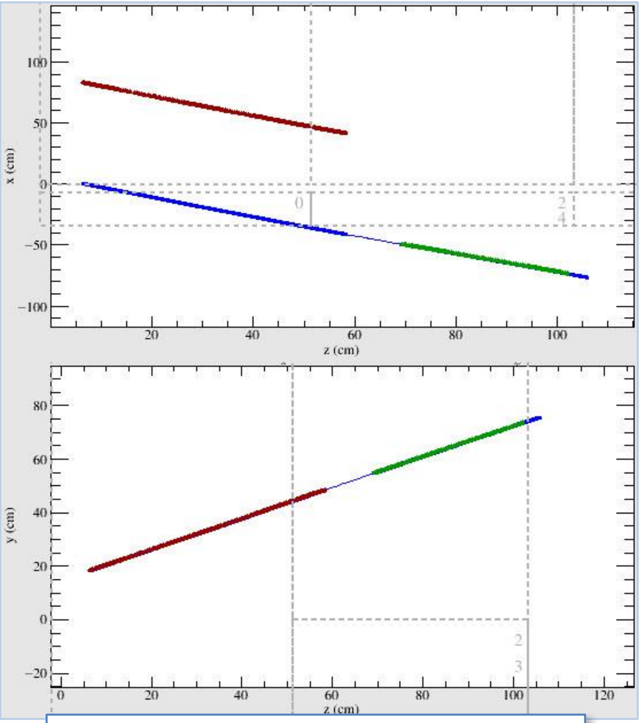
No stitching to show track parts reconstructed in each TPC.



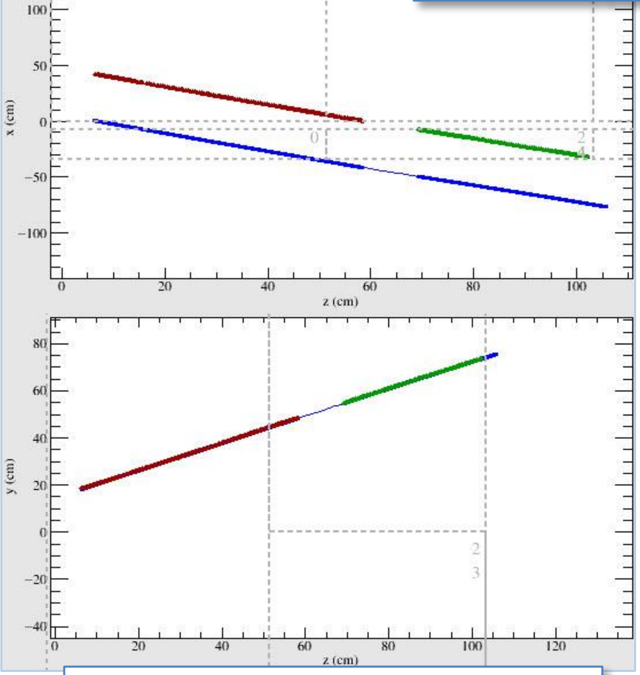
3D ref.points & stitching applied.

- Reconstruction of tracks exactly parallel to wire plane is the most difficult.
- Optimization can include 3D reference points:
  - e.g. track **endpoints** and **entry/exit points** are easy to find (***note:*** geometry divided into not too huge TPC volumes makes the reconstruction easier);
  - optimization is only *guided*: $(d_{ref\text{-}trk} - r)^2$ used as a distance to reference point measure
- There was a drop in reco efficiency for long muons parallel to wire planes, caused by failing stitching → now should be improved.
- Better isolated track reco → more accurate input to vertexing → more complex event topologies resolved.

APA crossing tracks with unknown T0



Reco result with random T0.



(don't worry about strange looking MC truth…)

Trajectories aligned, T0 found.

- As Karl suggested, look for similar endpoint distance to wire planes and co-linearity.
- Trajectory points shifted accordingly, tracks directions aligned, T0 associated with recob::Tracks
  - full track-vertex net shifted, so any deltas etc go together with muon
  - not merged as a single recob::Track to avoid strange effects in dE/dx calculation, easier access to study behaviour on APA in real data, or just apply fid.vol. cuts easier
  - probably PFParticle is a good place where two parts are logically connected
  - how to calculate time value for T0 from bare ticks count?
  - is it OK to associate T0 at this point of reco? is it useful, e.g. for multi-muon events to compare with T0s made from opt.flashes?
- Method can be confused if bad reco at endpoints, easy to push it more, depends on what we need for analysis.

Ongoing & TODO (general reco)

- Fight with EM cascades.
    - some flexibility in selection of EM-like tracks achieved: let's see what configuration works best for various purposes;
    - reduce time spent on reconstructing tracks in cascades
      – need to improve way fast rejecting wrong candidates
      – do not create vertices in EM-like regions
    - select dense EM parts on 2D level
- Technical change: put & optimize vertex inside the track, not only at endpoints – needed e.g. to avoid breaking muons by delta rays.
- …but what if the long track was a pion and the short track was e.g. proton: simple step-feature detector for dQ/dx sequence
- The same feature detector to find interaction & decay vertexes on along the track trajectory.

  (designed, partially implemented)

Data products for tracks in LArSoft

Current recob::Track

- contains vector of trajectory points and direction cosines
  - – tracks are used with association to recob::SpacePoints, a bit repeated information
- contains vector of dQ/dx vectors (one for each view, equal lengths)
  - – tracks have different lengths (no. of hits) in each view
  - – there are other modules producing dE/dx
- Assn to hits is used, but order of hits in assn is not ensured (however it works now – is it maybe luck?)

These issues were discussed few times with LArSoft team.

Recommendation:

- use metadata to store index with hit assigned to track
- use simple data product to keep extra info that is created for hits during track reconstruction

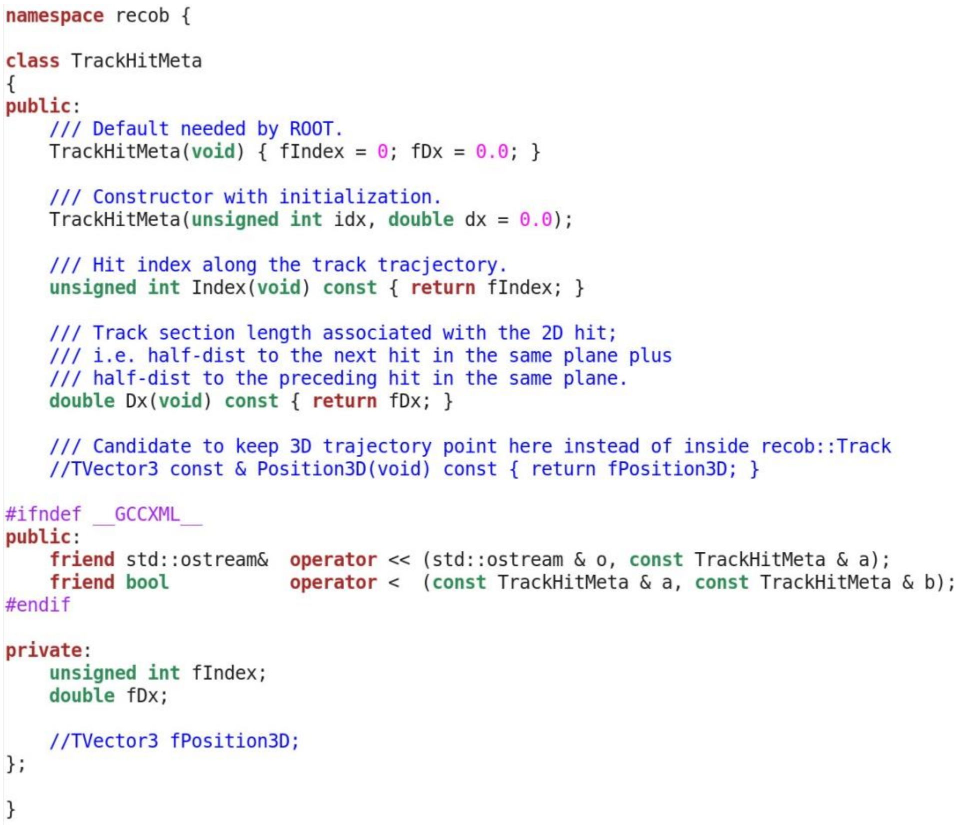
So the simple start is recob::TrackHitMeta class, so tracking modules can:

```
    produces< art::Assns<recob::Track, recob::Hit, recob::TrackHitMeta > >();
```

Still not perfect, but can be start point to organise a convenient (and common) structure.

Please, comment on it, suggest, … this is never a popular topic, but some effort would be needed to adopt code to use it and better to know that people agree on going in this direction.

Data products for tracks in LArSoft



```
namespace recob {

class TrackHitMeta
{
public:
    /// Default needed by ROOT.
    TrackHitMeta(void) { fIndex = 0; fDx = 0.0; }

    /// Constructor with initialization.
    TrackHitMeta(unsigned int idx, double dx = 0.0);

    /// Hit index along the track tracjectory.
    unsigned int Index(void) const { return fIndex; }

    /// Track section length associated with the 2D hit;
    /// i.e. half-dist to the next hit in the same plane plus
    /// half-dist to the preceding hit in the same plane.
    double Dx(void) const { return fDx; }

    /// Candidate to keep 3D trajectory point here instead of inside recob::Track
    //TVector3 const & Position3D(void) const { return fPosition3D; }

#ifndef __GCCXML__
public:
    friend std::ostream&  operator << (std::ostream & o, const TrackHitMeta & a);
    friend bool           operator <  (const TrackHitMeta & a, const TrackHitMeta & b);
#endif

private:
    unsigned int fIndex;
    double fDx;

    //TVector3 fPosition3D;
};

}
```

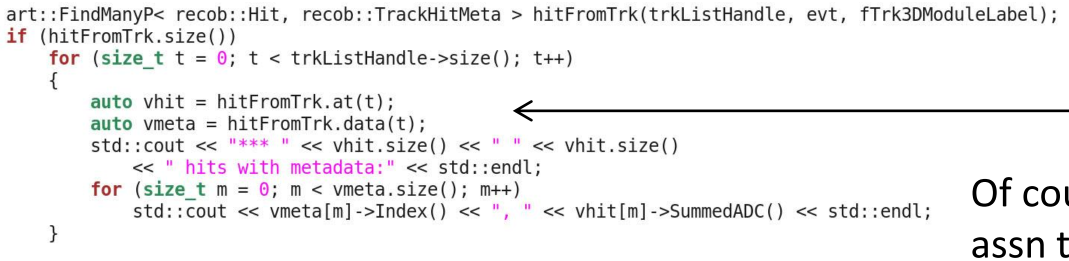
← in `recob` namespace, not `reco` as initially suggested (seemed more natural, but can be changed if not appropriate like this)

← can store more than dx, e.g. various steps of charge calibration (e lifetime, recombination, …), or even 3D position instead of SpacePoint

← makes sorting hits easy

would be good to have hits and metadata in a single vector instead of two parallel, but as you prefer

you can use like this:



```
art::FindManyP< recob::Hit, recob::TrackHitMeta > hitFromTrk(trkListHandle, evt, fTrk3DModuleLabel);
if (hitFromTrk.size())
    for (size_t t = 0; t < trkListHandle->size(); t++)
    {
        auto vhit = hitFromTrk.at(t);
        auto vmeta = hitFromTrk.data(t);
        std::cout << "*** " << vhit.size() << " " << vhit.size()
            << " hits with metadata:" << std::endl;
        for (size_t m = 0; m < vmeta.size(); m++)
            std::cout << vmeta[m]->Index() << ", " << vhit[m]->SummedADC() << std::endl;
    }
```

feature branches:
feature/rsulej_lardata_TrkAssnIdx,
feature/rsulej_larreco_TrkAssnIdx

Of course art will throw if you ask for the wrong assn type, so for the transition one may need to save both: the new <Track,Hit,TrkHitMeta> and the current <Track,Hit,void> …