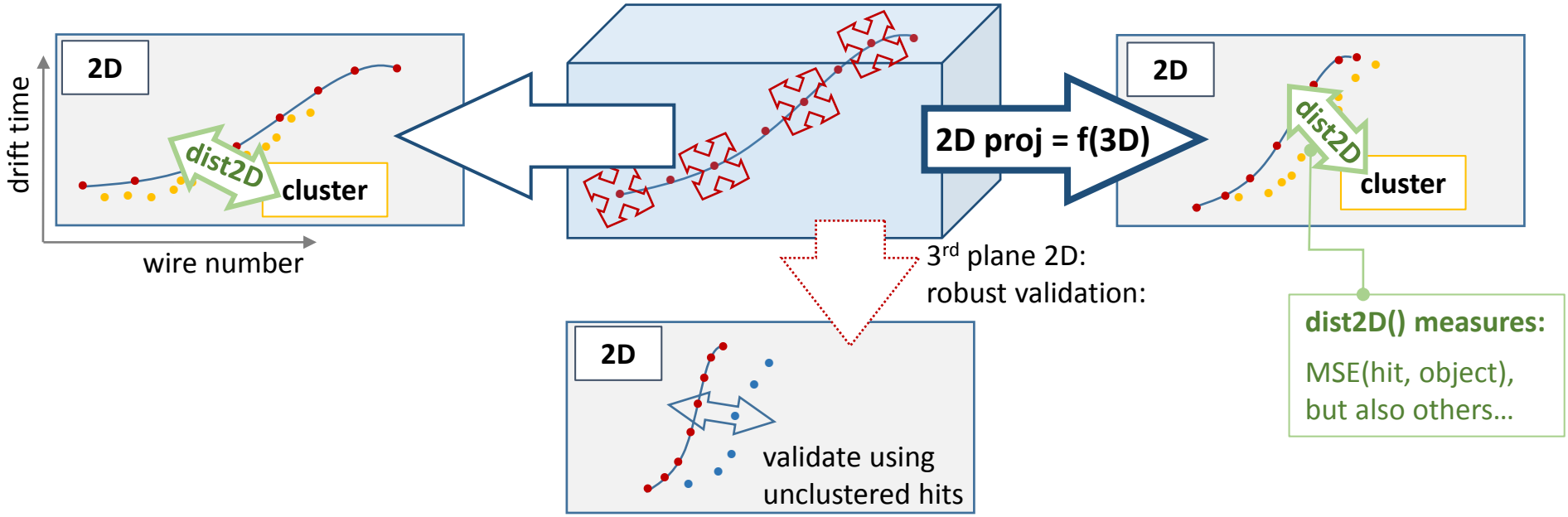
Projection Matching Algorithm
- updates -

Dorota Stefan, Robert Sulej
NCNR Warsaw, PL

LArSoft coordination meeting
29/07/2015

Projection Matching Algorithm

- works in 3D (on ***single track*** or ***full track structures***) to match the object's 2D projections to hits



- **new features in the module/algorithm**
- **algorithm parameters available through *fcl***
- **current work**

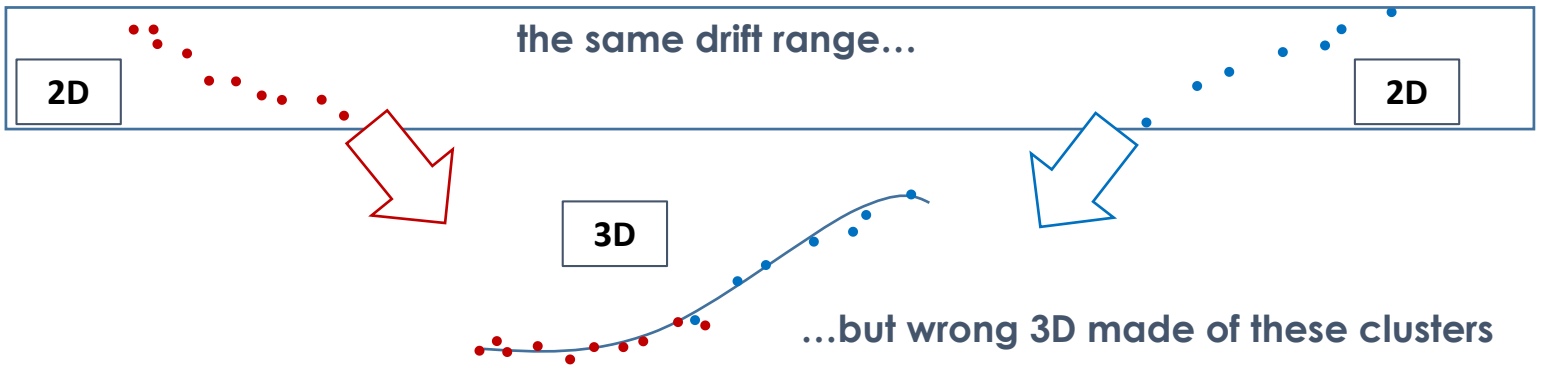
New features in the module

- **much better use of 2D clusters,** can select best 2D combinations (LineCluster as input)
  - start with largest cluster (#hits)
    - use cluster most overlapped in drift time
    - make 3D track candidate

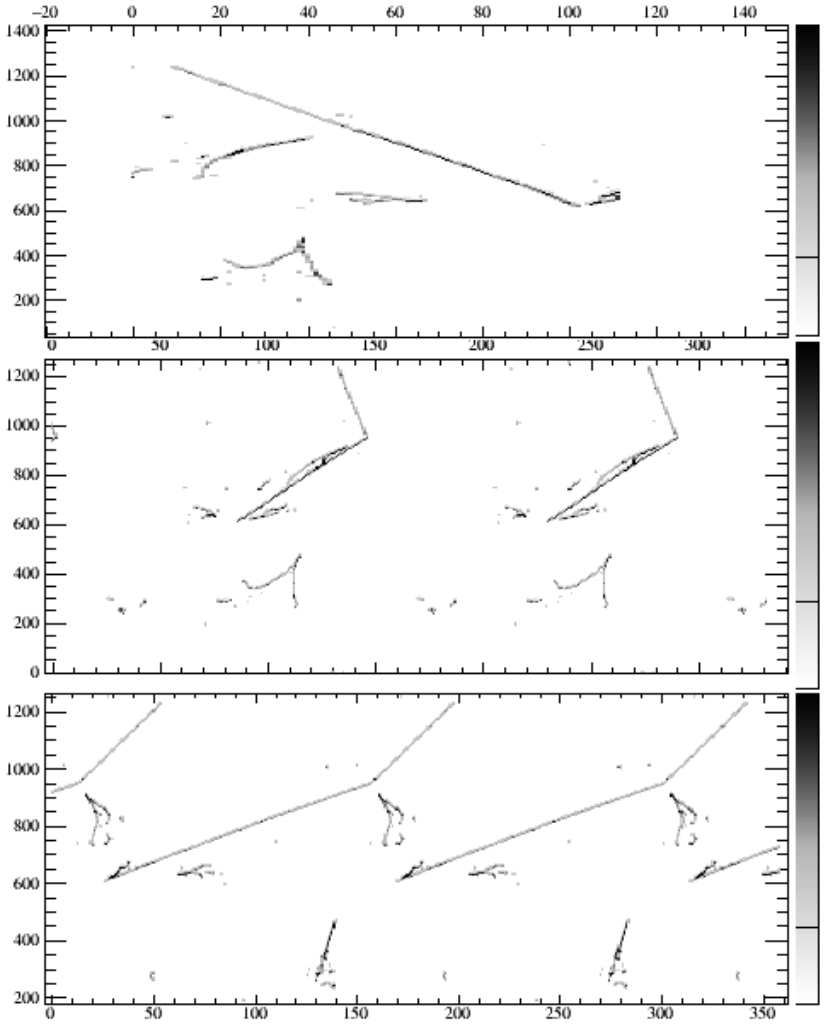
Tingjun and Tracy noticed low speed for MicroBooNE multi-muon events - many optimization are on the way

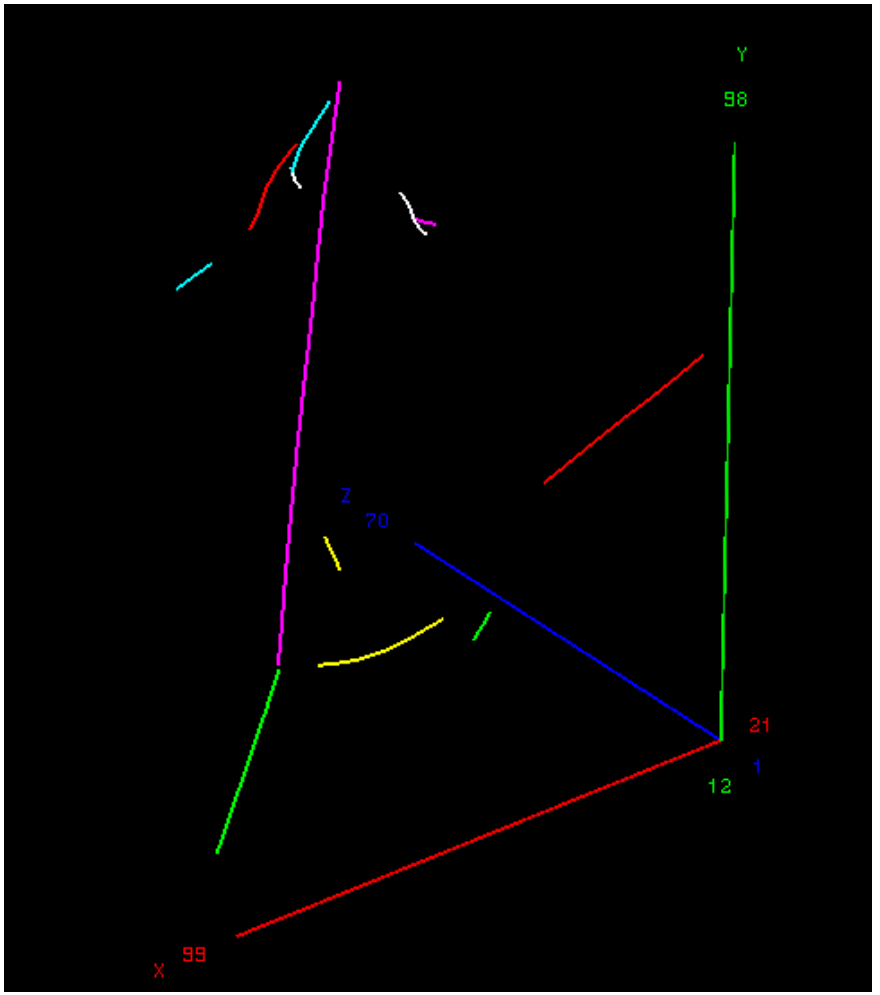
    - select best candidate (based on validation in 3rd view, MSE, fraction covered by intertwined hits from 2 views)
    - grow the track by adding clusters (partially) matching the trajectory
      - check validation measures
    - finally, add matching clusters from the validation plane
  - loop for large / then for small starting clusters
  - correct / merge / stitch / reoptimize …

- better behaviour in EM cascades and 2-plane geometry (thanks to Tingjun testing PMA on ArgoNeut data): quickly reject track candidate if 2D hits are not intertwined enough along 3D track



1.5GeV π^- interaction with two π^0s, LineCluster as an input.

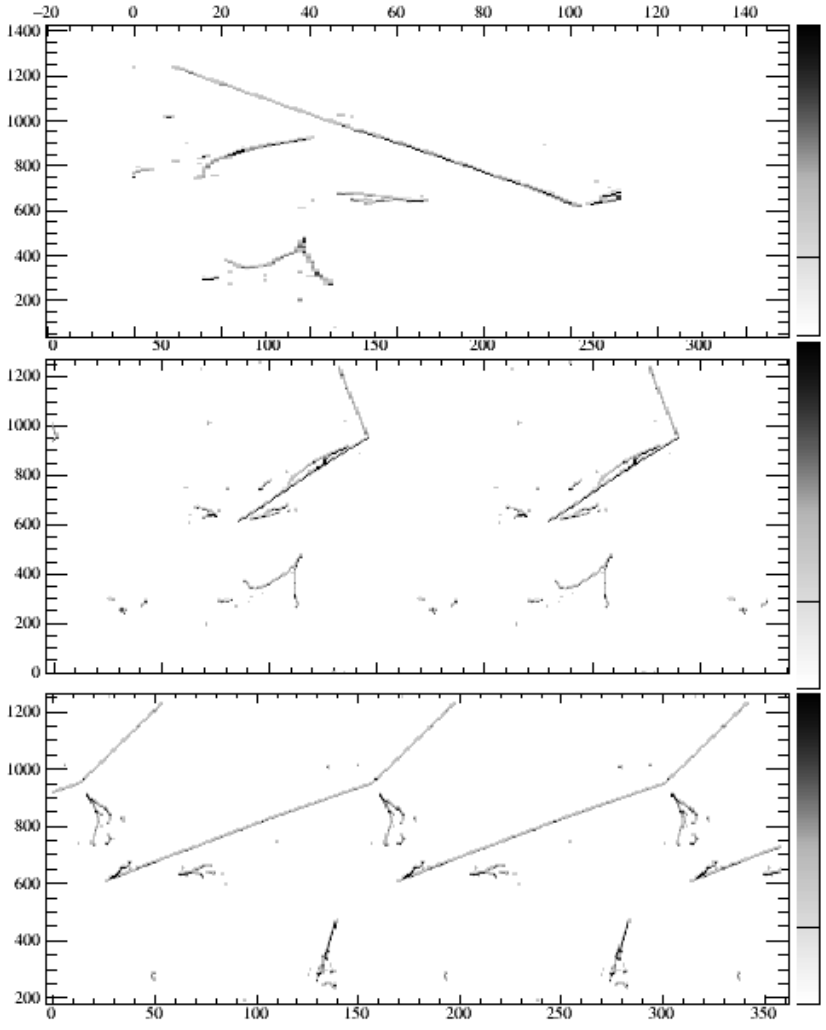


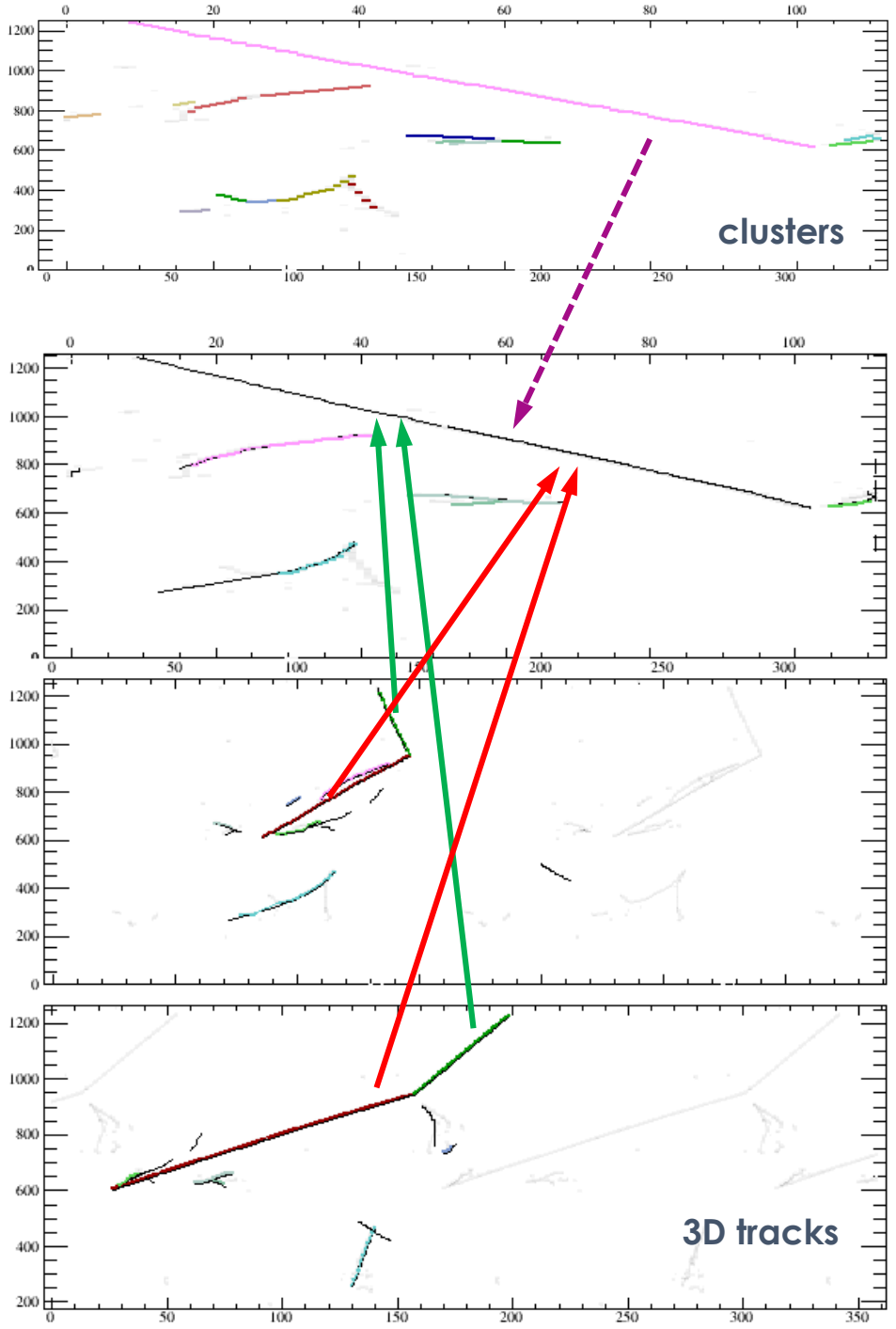


Hadron (μ as well) tracks seem to be very efficient, some evident tracks found in **low energy EM cascades** thanks to ClusterCrawler, spurious tracks not frequent – again, this is low energy.

example of 2D cluster not matched with 3D tracks

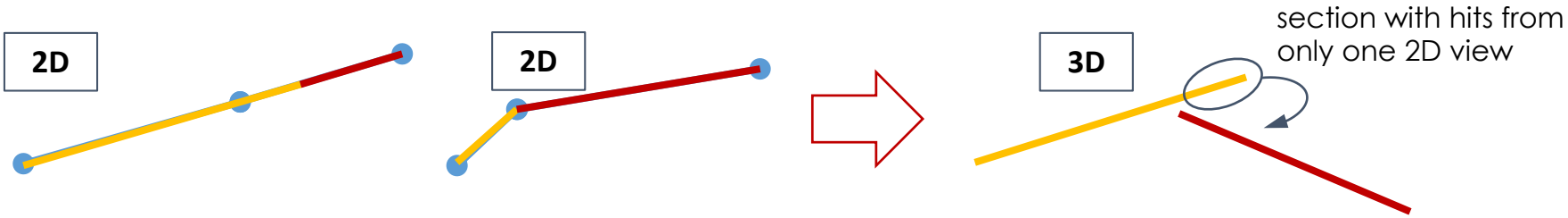
…prefer to implement vertices first instead of adding dedicated if-else



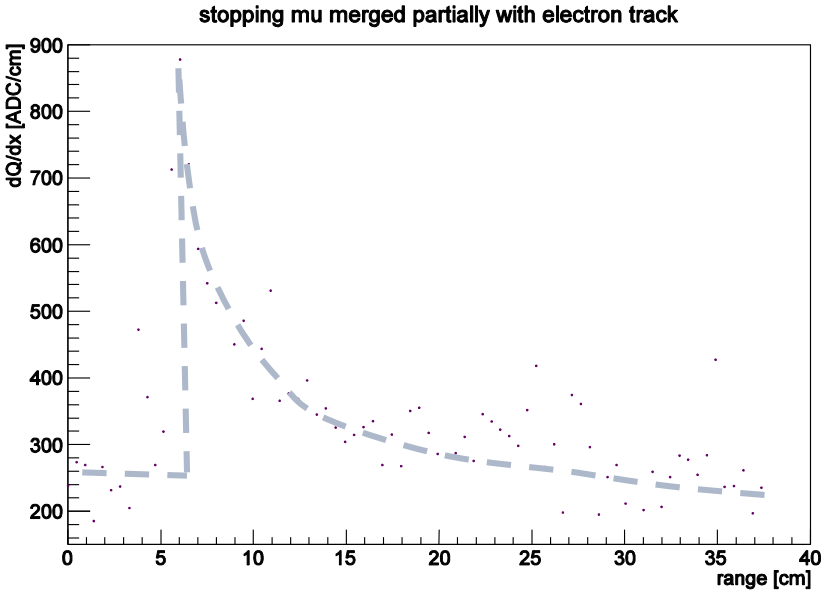


New features in the module

- correct hit-track assignment caused by linear projection of kinks in one of 2Ds



if tracks colinear in both 2D views then correction of similar mistakes needs to look at e.g. dQ/dx (one of things to implement…)

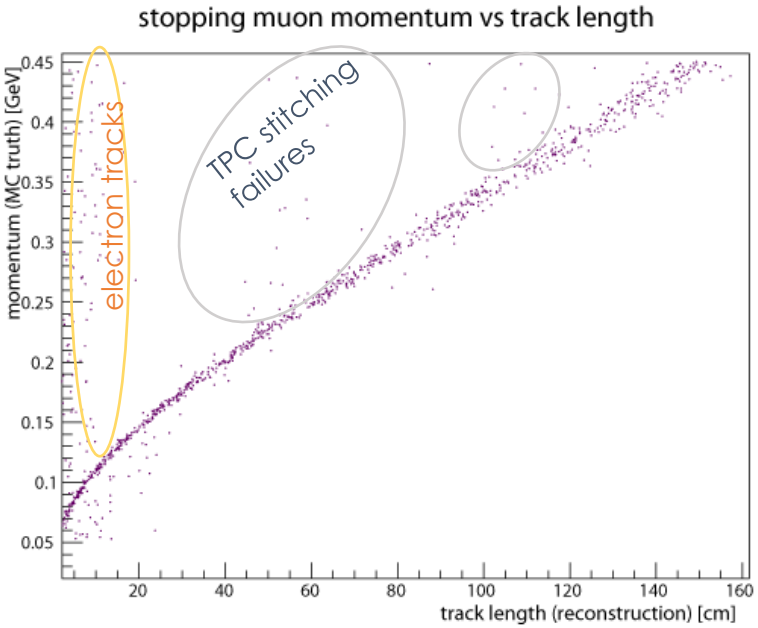


stopping μ partially merged with the electron track

New features in the module

- stitching colinear tracks between TPCs (*default: on*), merging tracks within a TPC (*default: off*), optimization of trajectory that is crossing many TPCs

  track is reoptimized after stitching/merging to obtain smooth trajectory



stopping μ's with the initial momentum 50-450 MeV/c,

crossing up to 3 TPCs

some cases failed – to be checked

```
MergeWithinTPC:        false   # merge witnin single TPC; finds tracks best matching by angle:
MergeTransverseShift:  2.0     #    - max. transverse displacement [cm] between tracks
MergeAngle:            0.5     #    - max. angle [degree] between tracks (nearest segments)
                               #
StitchBetweenTPCs:     true    # stitch between TPCs; finds tracks best matching by angle:
StitchDistToWall:      3.0     #    - max. track endpoint distance [cm] to TPC boundary
StitchTransverseShift:3.0      #    - max. transverse displacement [cm] between tracks
StitchAngle:           7.0     #    - max. angle [degree] between tracks (nearest segments)
```

New features in the module

- flip track to increasing Z (*beam*), decreasing Y (*down, CR*), increasing dQ/dx (*stopping*, overrides Z and Y flips if dQ/dx rise is significant)

```
FlipToBeam:     false  # set the track direction to increasing Z values
FlipDownward:   false  # set the track direction to decreasing Y values
AutoFlip_dQdx: false  # set the track direction to increasing dQ/dx
```

- many fixes & improvements due to our LArSoft missunderstanding + a lot of help, hints, tests from Tingjun and Tracy

speed/precision

may be useful for different wire pitch / orientation and different track multiplicity

standard_pmalgtrackmaker:

```
OptimizationEps: 0.01    # rel.f chg that stops optimization after adding a node
FineTuningEps:   0.0001  # rel.f chg which stops fine-tuning of optimized track

TrkValidationDist2D:1.0   # max.dist. [cm] in the validation in the "third" plane
HitTestingDist2D:   0.5   # max.dist. [cm] in testing compatibility of hits
MinTwoViewFraction: 0.4   # min. fraction covered with hits from many 2D views

HitWeightZ:  1.0  # weights used for hits in U, V, Z planes:
HitWeightV:  1.0  #    - lower values for planes where hit position
HitWeightU:  1.0  #    is less reliable (e.g. due to S/N), sum does
                  #    not need to be 1.0
```

to change the impact of hits on the objective function

New features in the algorithm

- build straight segment: **EM cascade axis** or its **initial, track-like part**

```
pma::Track3D* buildSegment(
   const std::vector< art::Ptr<recob::Hit> >& hits_1,
   const std::vector< art::Ptr<recob::Hit> >& hits_2) const;
```

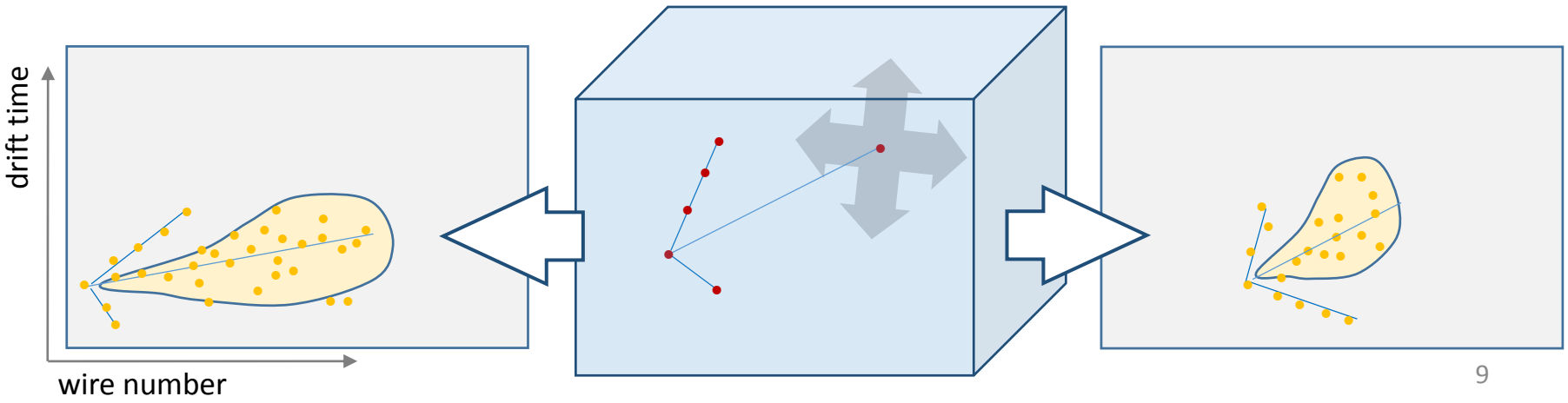
hits from two 2D views, 3D vertex **not known**

```
pma::Track3D* buildSegment(
   const std::vector< art::Ptr<recob::Hit> >& hits_1,
   const std::vector< art::Ptr<recob::Hit> >& hits_2,
   const TVector3& point) const;
```

hits from two 2D views, 3D vertex **known and fixed** in PMA segment optimization

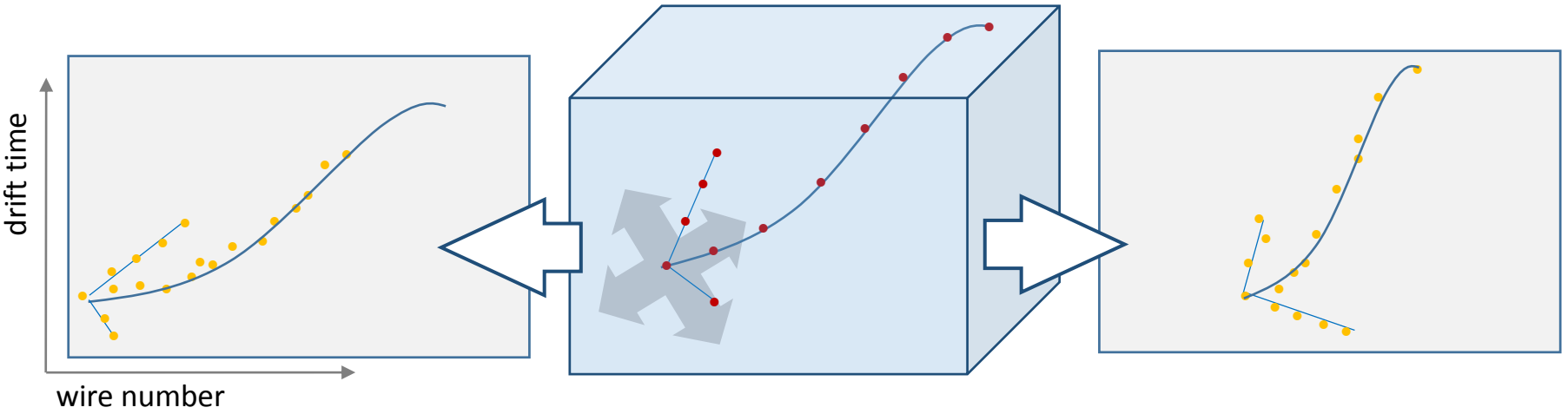
- **dQ/dx in the initial part of EM cascade**
  - select hits close to the 2D projection of fitted segment
  - start selection from the segment front, skip the first hit (poor dx estimation),
  - last selected hit < 2.5cm from the front or last before significant charge increase (showering start)
  - exact dx calculated from section of the 3D segment corresponding to selected hits

```
double selectInitialHits(pma::Track3D& trk, unsigned int view = geo::kZ) const;
```



Current work

- mutli track structures → **vertex** with improved **position**, **track directions**

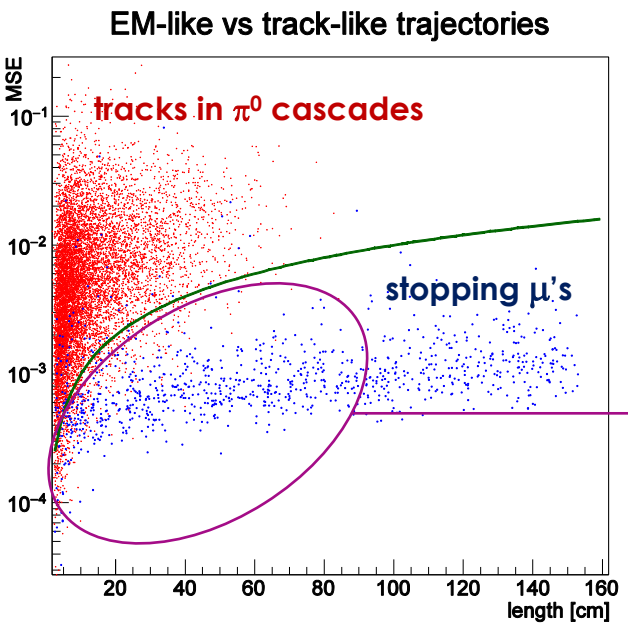


- PMA nodes, track, etc: most of the functionality needed to handle track connections and reoptimize the whole net of tracks is there
- need to implement finding of vertices (resume earlier work, that was actually quite advanced)

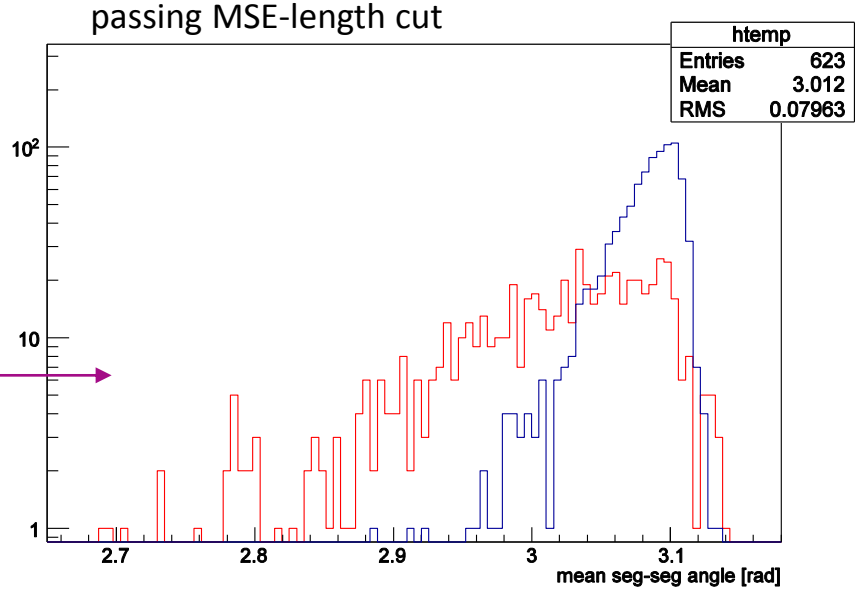
- need to refactor PMA module since it is growing too big, probably feature branch is a good idea

Current work

- **electron/EM-cascade-part** versus **hadron/muon track**



mean angle between segments for tracks passing MSE-length cut



- first trials, still testing, **may not be the solution for large showers**…
- MSE / curvature measures are not completely enough, would like to try rough checks of dQ/dx
- could subtract tracks and produce container of EM-cascade-hits (*low energy*)
  - can be done within the module
  - or maybe better: only tag the track and let create container in another module, if needed – now the „tag" is encoded in the recob::Track index, is there a better way?

Summary

- PMA module developed enough to be practically applied
  - efficient use of input clusters

    results depend on clustering efficiency, however we try to provide generic processing that may work with various clustering algorithms

  - stitching needed for multi-TPC ready, of course we will investigate failures

  - tested on ArgoNeut data, tried on MicroBooNE simulation: many bug fixes made, **speed improvements** on the way

- Multi-track structures and vertex oprimization is the priority in the next developments.

- Number of possible improvements was spotted – to be implemented.

- Visualisation (*just to remember to ask*): what could be a good way to show in 3D which plane hit comes from, how to show track nodes (assign vertices? but this is only to help development, not data product)

- We appreciate very much tests on various geometries / data conditions, such feedback is invaluable!