Clupatra

Topological TPC pattern recognition

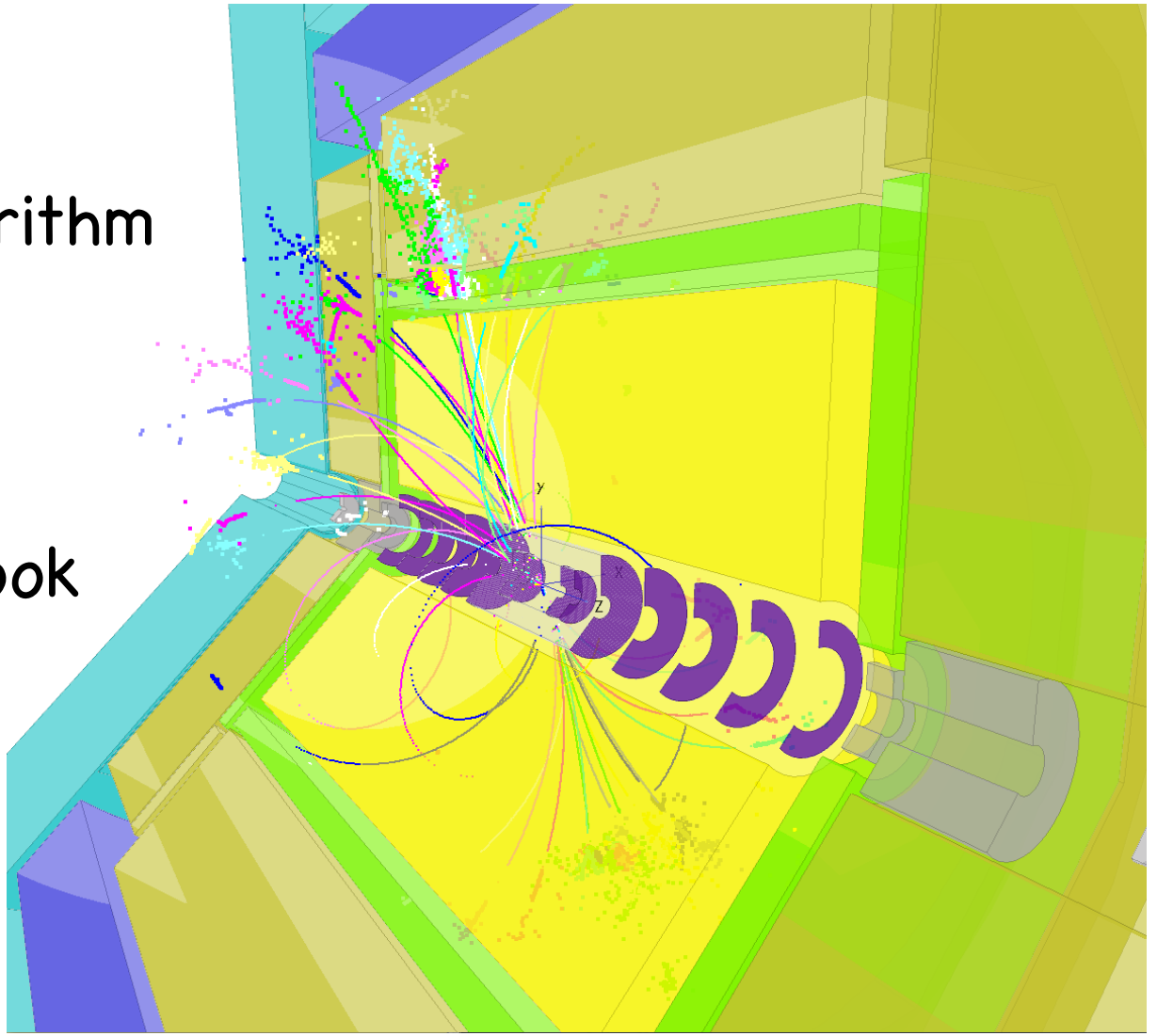
Frank Gaede, DESY

LCWS 2011

Granada, Spain, Sep 26-30, 2011

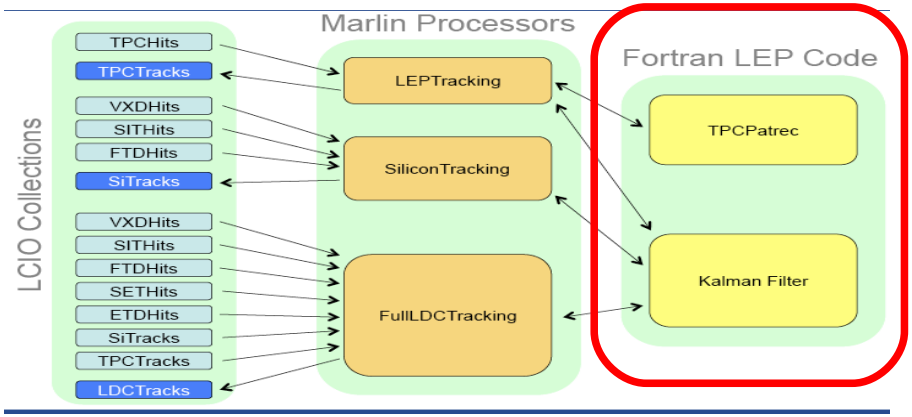
Outline

- Introduction
- the clupatra algorithm
- performance
- ClupatraNew
- Summary & Outlook



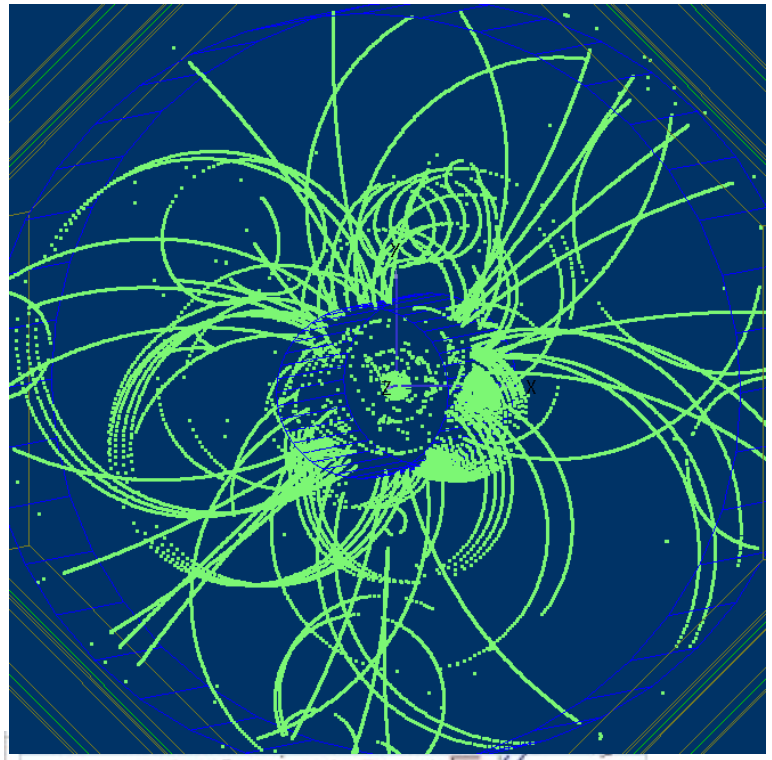
Introduction

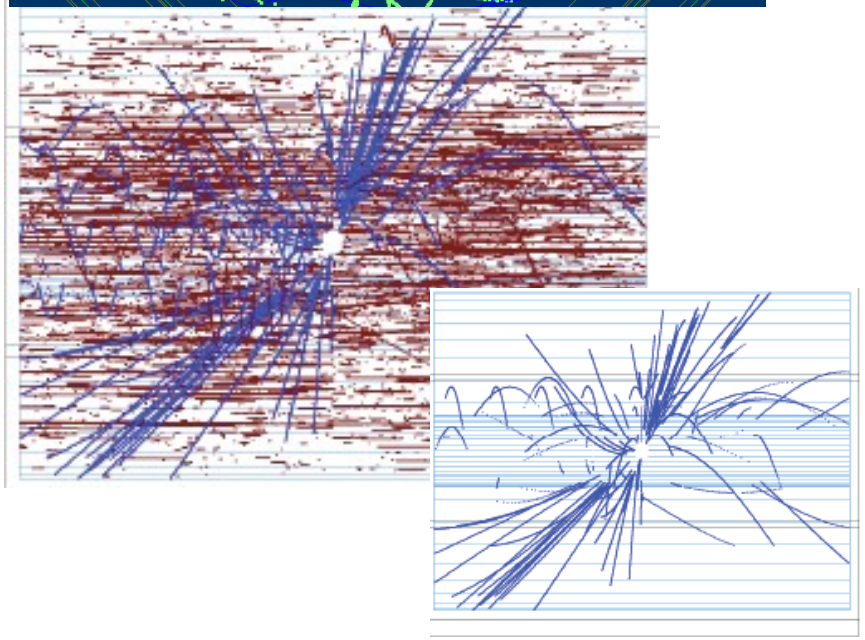
- for ILD we identified the need to replace old f77tracking code in order to improve the sw maintenance and the performance (background studies, 1 TeV)



- for this we need:
- a new C++ Kalman filter tool
  - chose KalTest/KalDet (K.Fujii et al)
  - included in iLCSoft v01-10
  - also used by LCTPC/MarlinTPC
- adopt SiTracking to new Kalman fitter
  - possibly improve/develop new algorithm (Fwd !)
- rewrite the TPC pattern recognition <- this talk

TPC Pattern recognition





- patrec in a TPC should be rather easy
  - tracks immediately visible
  - "could be done by a kid with crayons"
- ILD TPC has a huge number of voxels >220 hits on most tracks
- classic triplet search and combinatorial Kalman filter probably overkill (CPU & coding intensive)
- distance between hits on given track is typically much smaller than distance between tracks
- => can use NN-Clustering
- micro curlers from pair bg should be removed beforehand (->LOI)

Template for NN-clustering algorithm

simplest *nearest neighbor clustering:*
- loop over all hit pairs
- merge hits into one cluster if *d(h1, h2) < cut*
- d() could be 3D-distance
  - also anything else (trk params)

```
class NNClusterer{

public:
  typedef T value_type ;
  typedef Cluster<T> cluster_type ;
  typedef Element<T> element_type ;
  typedef PtrVector< element_type >    element_vector ;
  typedef PtrVector< cluster_type >    cluster_vector ;
  typedef PtrList< element_type >      element_list ;
  typedef PtrList< cluster_type >      cluster_list ;

  /** Simple nearest neighbour (NN) clustering algorithm. Users have to provide an input iterator of
   *  Element objects and an output iterator for the clusters found. The predicate has to have
   *  a method with the following signature: bool operator()( const Element<T>*, const Element<T>*).
   *  All pairs of elements for which this method returns 'true' will be merged into one output cluster
   *  - all other pairs of elements will be in different clusters.
   */

  template <class In, class Out, class Pred >
  void cluster( In first, In last, Out result, Pred& pred , const unsigned minSize=1) {
```

- generic NN-Clustering template
  - works for any C++ class
  - uses std::lists for clusters
- main classes
  - Cluster< MyClass > , Element< MyClass >
- use predicate class for cut
- STL like code:

useful utility
including containers
and helper classes
-> could release in **ilcutil**

```
nncl.cluster( hits.begin(), hits.end(), std::back_inserter( trkclu ), HitDistance(40.) );
```

Clupatra step 1

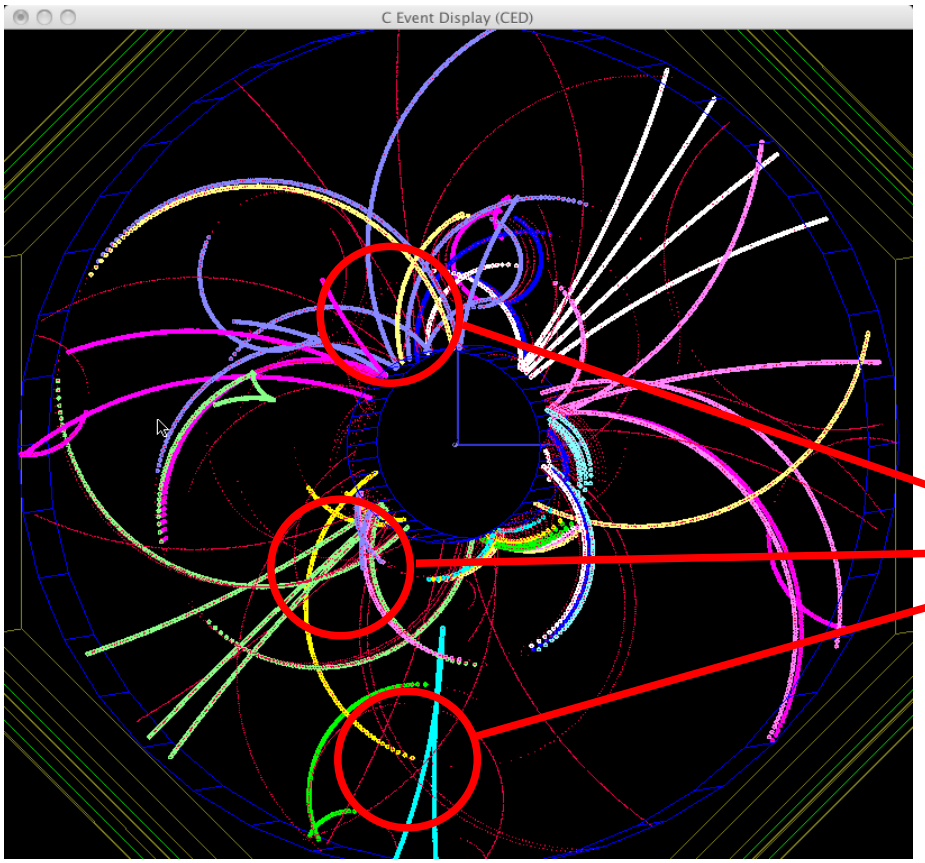
- **nearest neighbor clustering**
- use simple euclidian distance
  - d = sqrt(dx^2 + dy^2 + dz^2) < C ; C = 40mm
- use z-index + sliding window to speed up processing

```
struct HitDistance{  float _dCutSquared ;
  HitDistance(float dCut) : _dCutSquared( dCut*dCut ) {}

  inline bool operator()( Hit* h0, Hit* h1){

    if( h0->first->layer == h1->first->layer )
      return false ;

    return ( h0->first->pos - h1->first->pos).r2()
      < _dCutSquared ;
  }
} ;
```

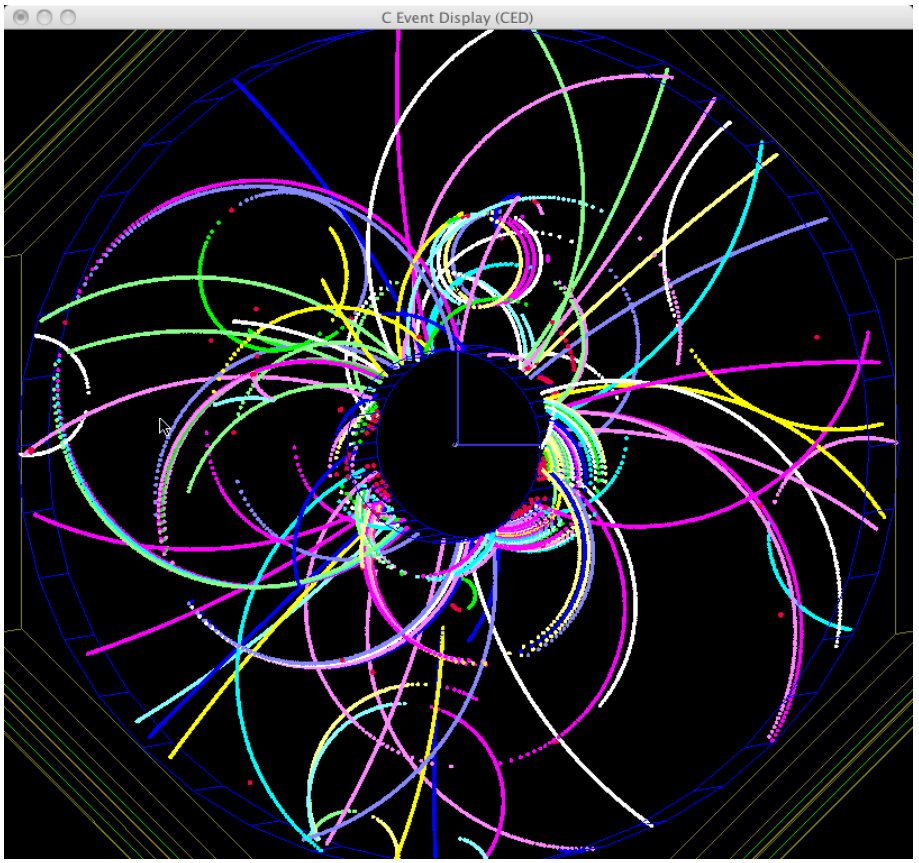


example:
- ttbar event @ 500 GeV

obvious issue:
- close by tracks are merged into one cluster

Clupatra step 2

- for merged clusters (duplicate pad row fraction):
  - **cluster in pad row ranges** (e.g. 15 rows) **– going inwards**
  - identify **clean track stubs**
  - **extend clean stubs forward & backward using Kalman fitter**
    - add best matching Hit if delta(chi2) < 35.
    - update track state !

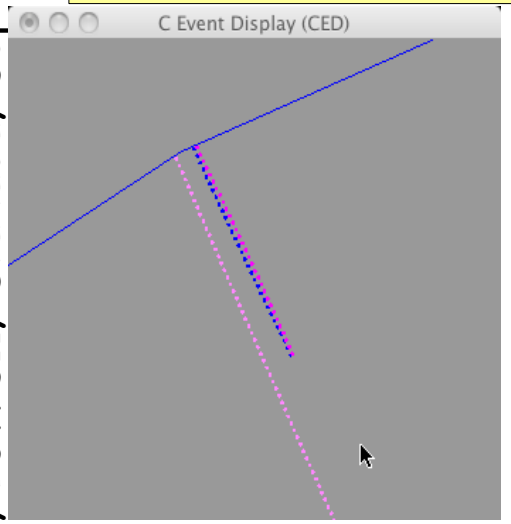


example:

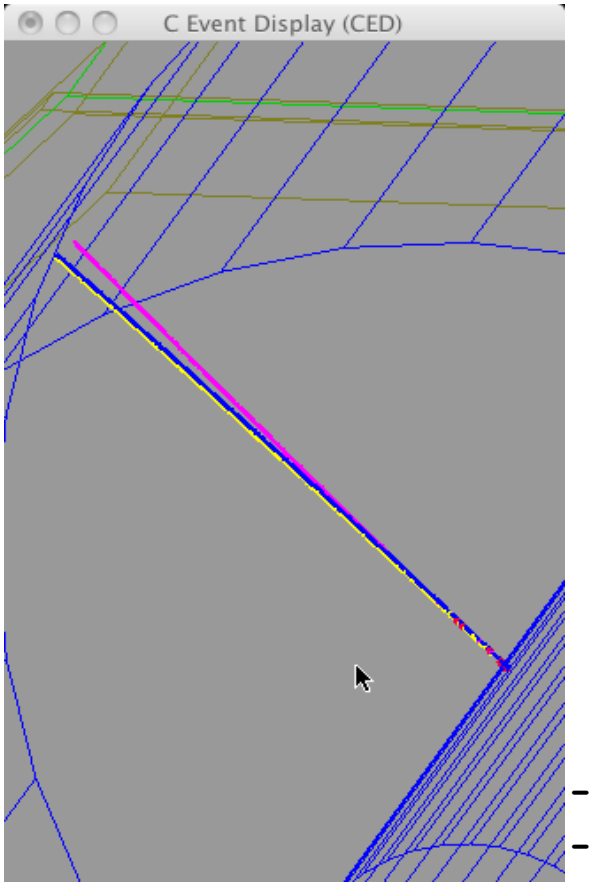
- ttbar event @ 500 GeV
- results in clean tracks and segments for curlers
- little leftover hits (red)
- some very close by tracks lost (taus/conversions)

Clupatra step 3

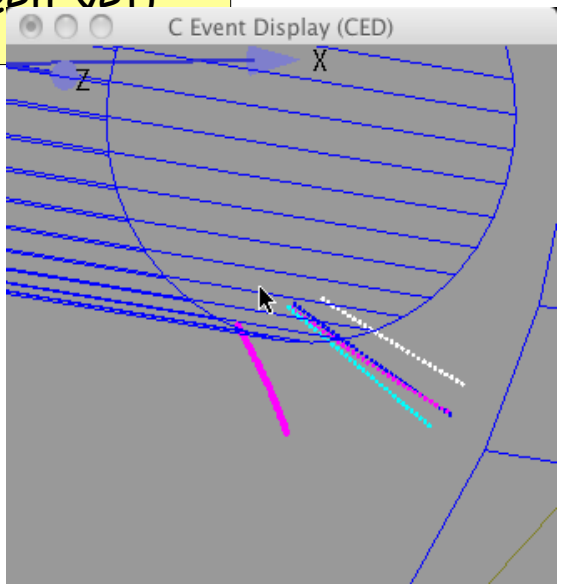
- re-cluster in leftover hits (NN clustering)
- based on pad row multiplicity force into one, two or three clusters
- apply KalTest fit to throw out falsely merged hits (rare)
  - higher multiplicity to be done (extremely rare - not seen yet)



- gamma conversion in barrel
- forced into two tracks



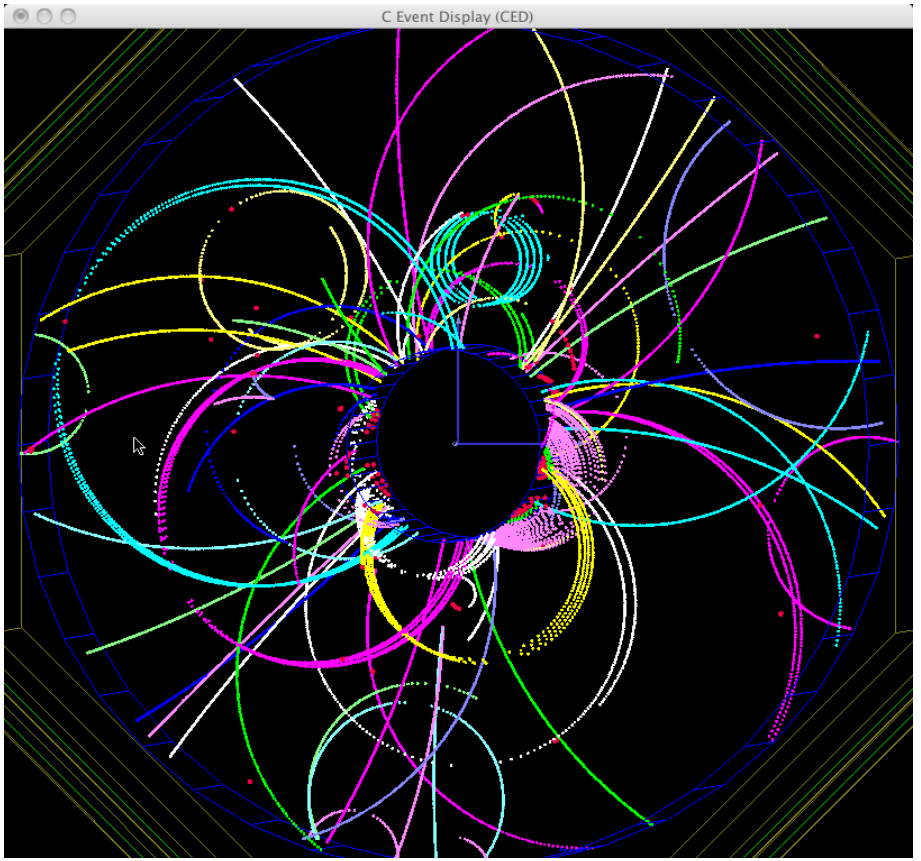
- three prong tau - barrel
- two close-by tracks forced into two tracks



- five prong tau - forward
- three close-by tracks forced into three tracks

Clupatra step 4

- **merge track segments (from curlers)**
- based on rough (O(10%)) criterion for R, delta(xc,yc), tan(lambda)
- disallow overlaps in z

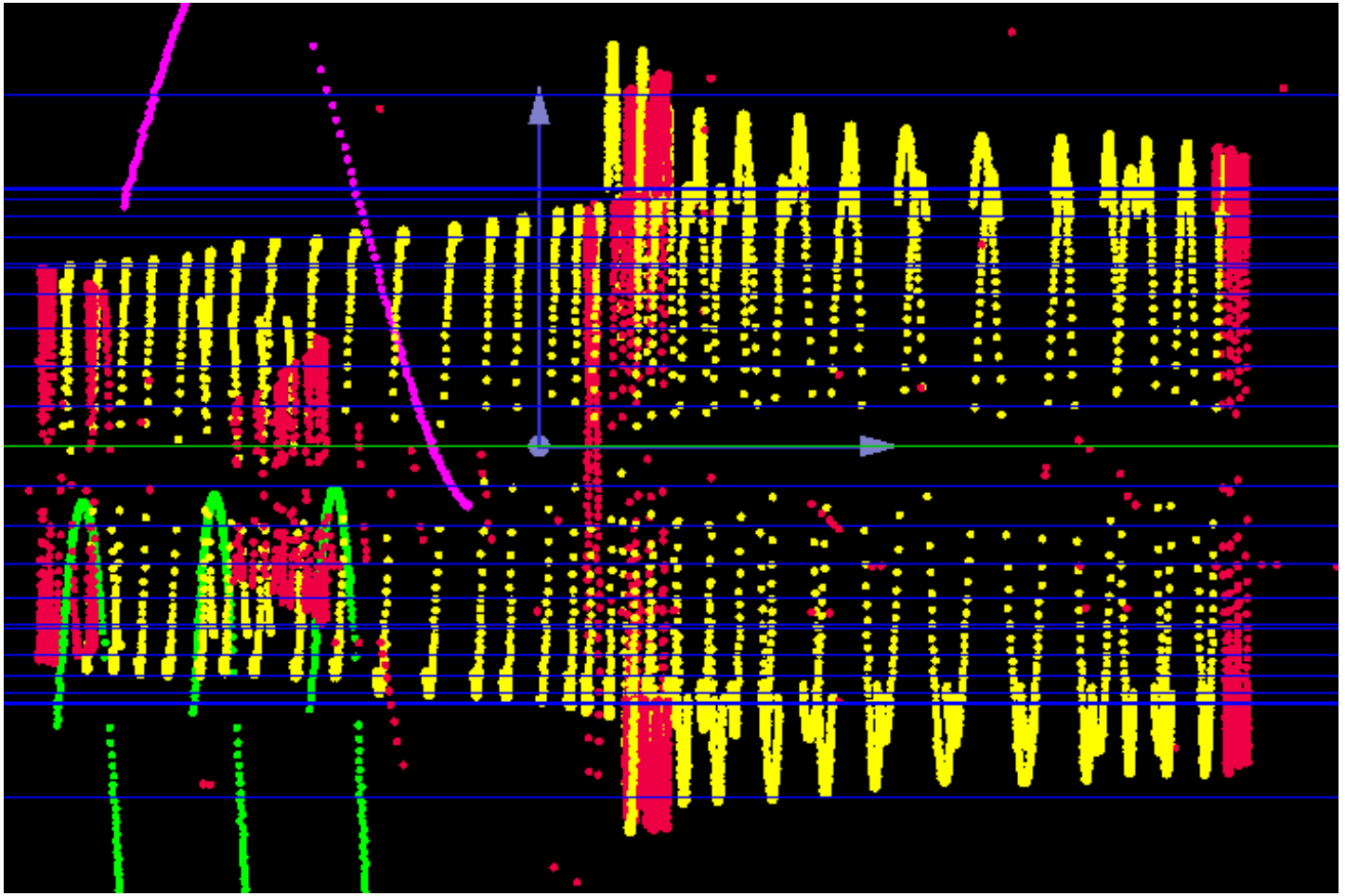


example:

- ttbar event @ 500 GeV

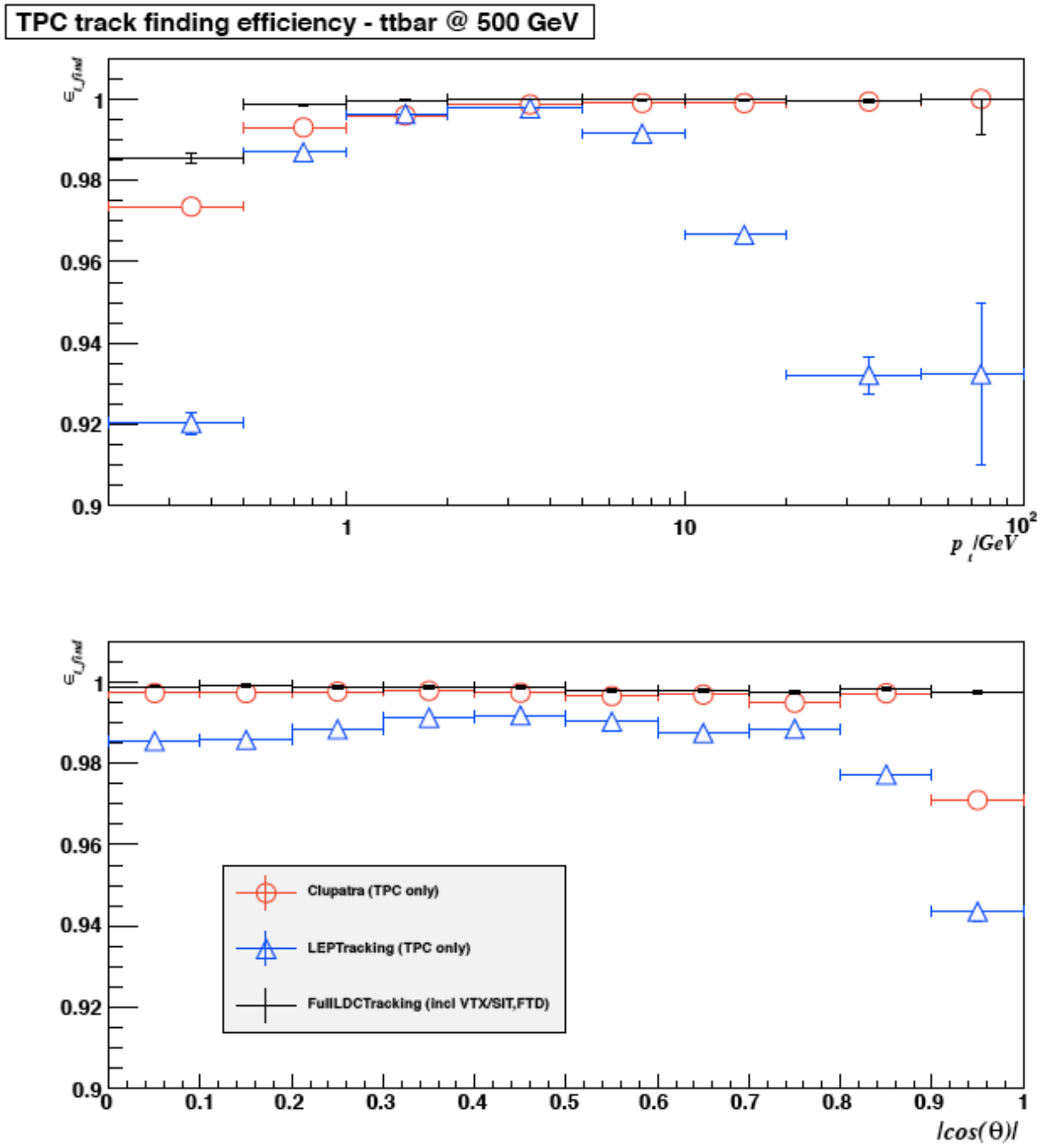
- works nicely
- few segments are not merged
- most of these curler segments where lost in old patrec

Limits of the algorithm



- Clupatra finds interleaved curlers - to large extend
  - yellow: clupatra track - red: hits have been missed
- this muon curls back into itself five times !
- don't need to deal with this often :-)

track finding efficiency I



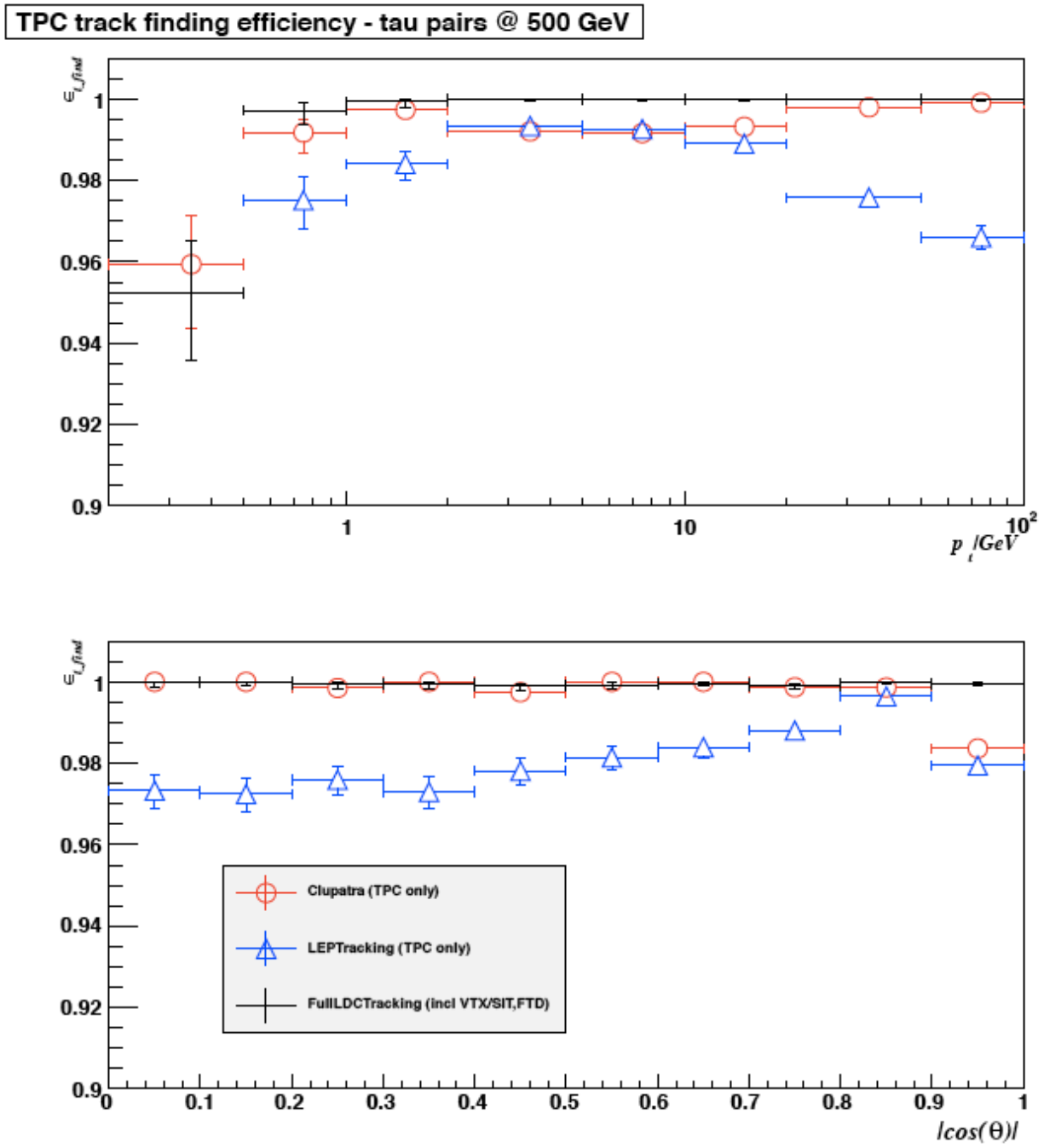
- prompt tracks PCA(IP)<10cm
- > 5 TPC Hits
  - (pt >100 MeV)
  - (|cos(th) |>.99)

- comparison to LEPTracking pattern recognition

- NB: Clupatra has no fully reconstructed tracks yet and no quality cuts are applied

- high efficiency demonstrates that algorithm works and could replace old f77 code soon

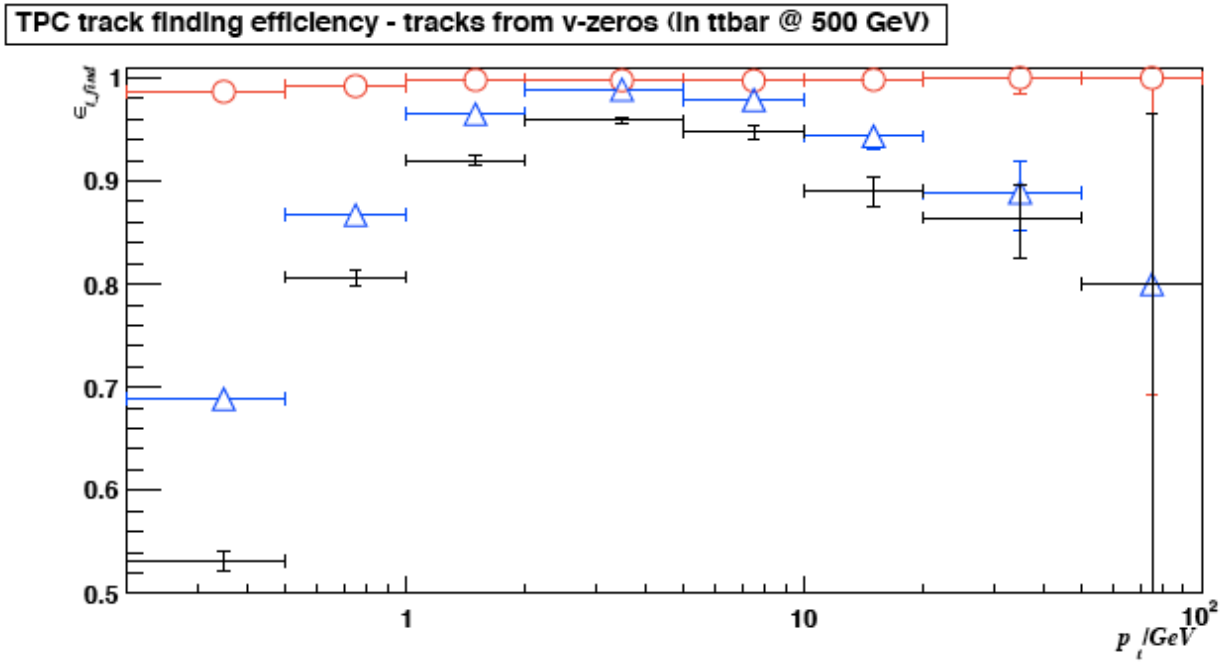
track finding efficiency II



prompt tracks PCA(IP)<10cm
> 5 TPC Hits

- (pt >100 MeV)
- (|cos(th) |>.99)

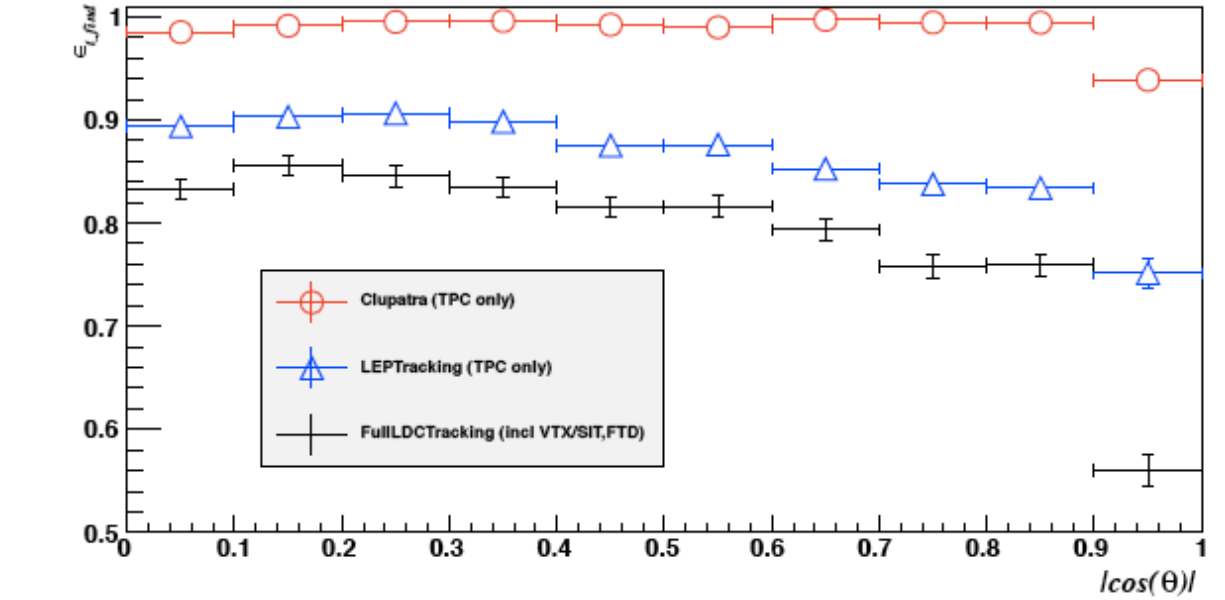
track finding efficiency III



non-prompt tracks
- rho_vtx > 10cm

parent charge==0
- mostly vzeros and conversions

> 5 TPC Hits
- (pt >100 MeV)
- (|cos(th) |>.99)



=> will gain efficiency for reconstruction of V0s / conversions

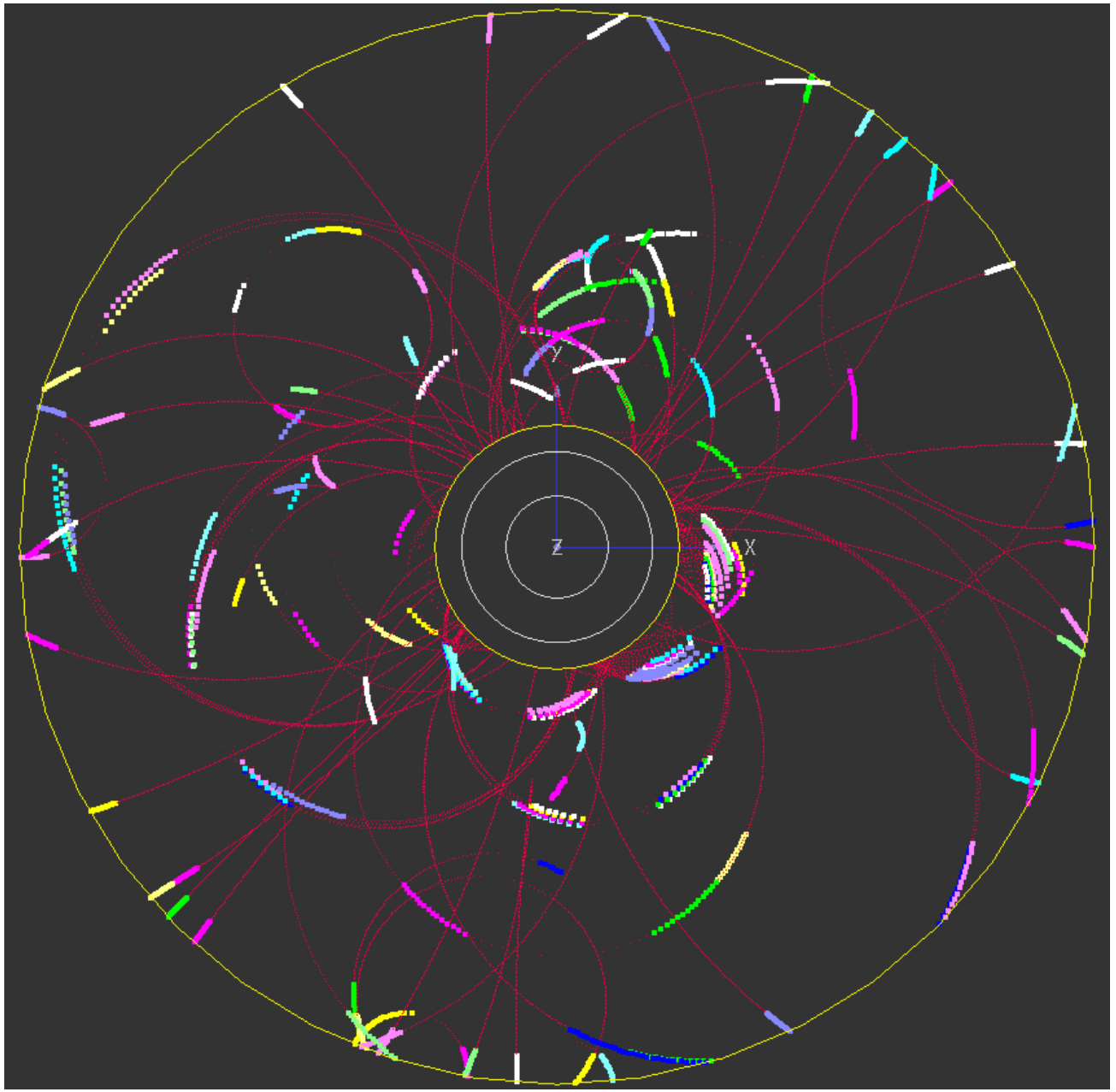
"ClupatraNew"

- results shown with initial version of Clupatra (v00-01)
- since then almost complete re-write:
  - cleaned up code and algorithm
  - re-factored NN-Clustering
  - better memory management (containers w/ ownership)
  - removed step 1 : intial NN-clustering w/ all hits

    -> start with small clean cluster seeds

    -> faster w/ same or better performance
  - switched to use new implementation of new **IMarlinTrk**
  - released as v00-02

Clupatra seed cluster

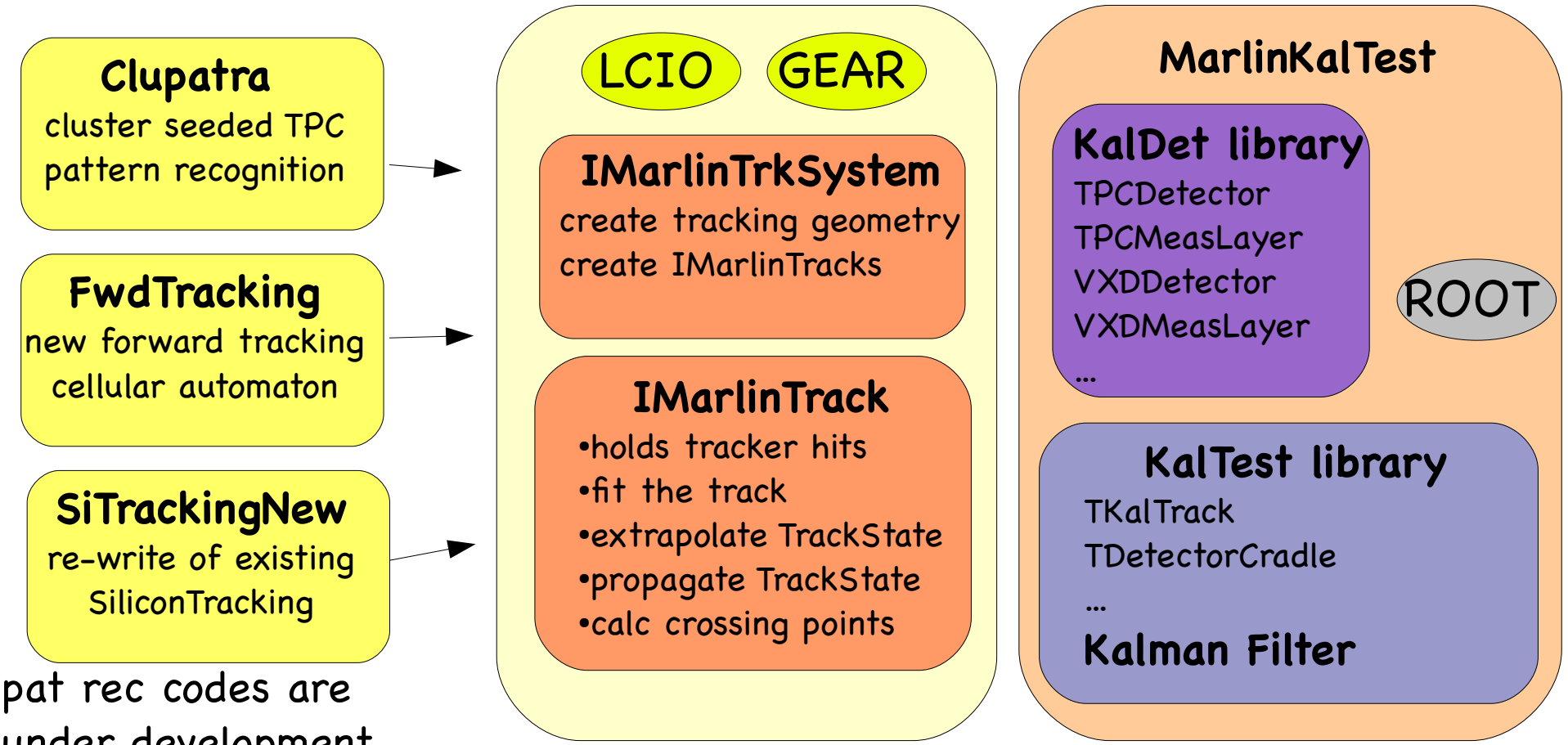


- starting with seed clusters speeds up the algorithm
- clustering ~N^2
- filtering ~N

- on my laptop:
  - old 5s/ttbar evt
  - new 3s/ttbar evt
  - (f77: ~1s !)

IMarlinTrk interface

- need common framework for developing new tracking code (TPC, Silicon, Fwd)
- would like to de-couple patrec and fitting
- defined abstract interface **IMarlinTrk** and implement using KalTest/KalDet
  - other fitters might follow (GenFit, ...)
- serves as tests case for writing a generic tracking package in AIDA WP2



Summary & Outlook

- a new topological TPC pat-rec has been developed combining clustering and combinatorial Kalman filter methods
- first results show improvements compared to existing LEPTracking pat-rec code
- recently rewritten to use new MarlinTrk package
- Outlook – To Do
  - write documentation
  - work on track quality cuts
  - study effects of background
    - pair-bg and gamma-gamma
  - extend tracks to pick up hits in inner Si-trackers: VXD, SIT and FTD