Clupatra tracking for Test Beam data

Oleksandr Volynets

DESY

Apr 04, 2013

Outline

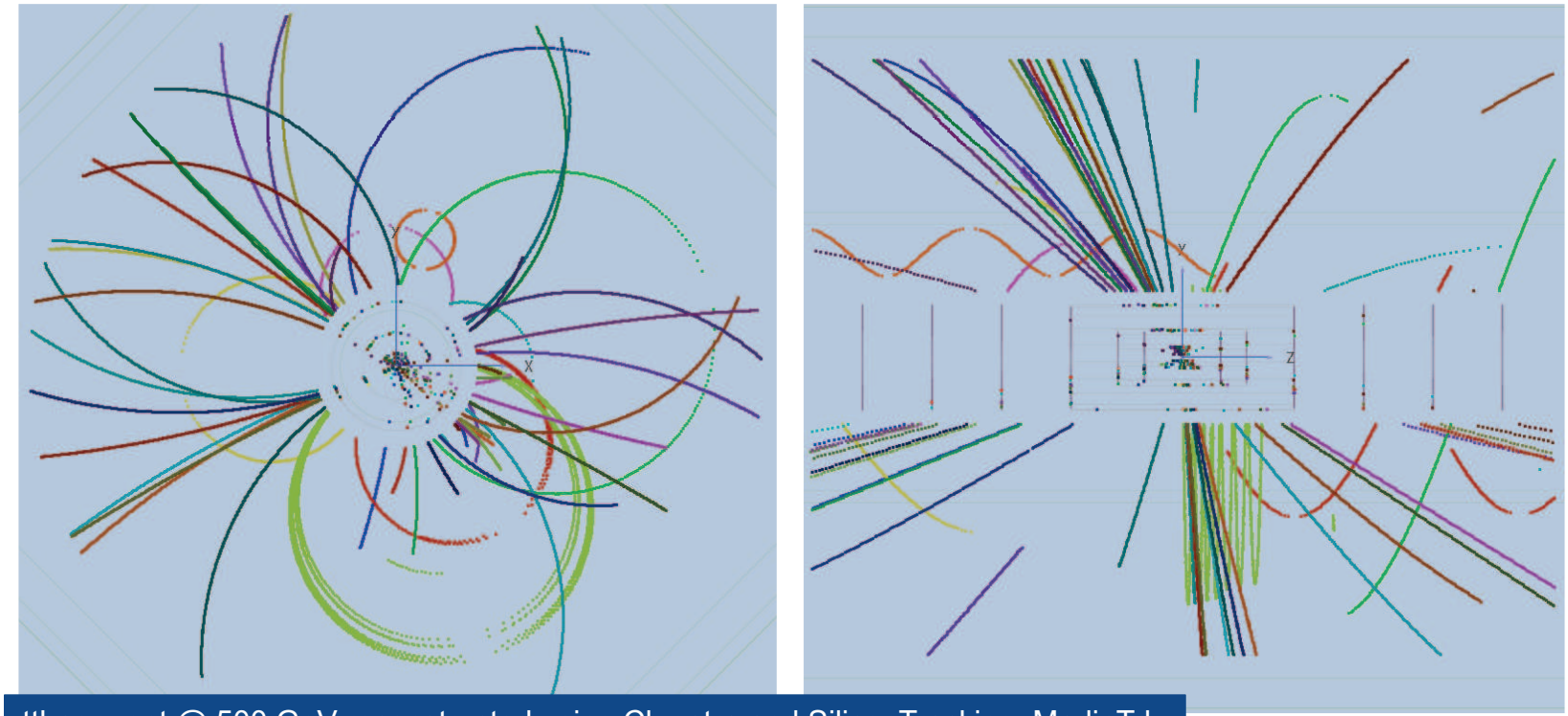
- Introduction
- Clupatra dependencies
- Pattern recognition algorithm
- Examples of the track fits in test beam data
- Summary

Introduction

Clupatra ...

- is a Pattern Recognition package designed for *ILD TPC* tracking;

Complete Track Reconstruction



ttbar event @ 500 GeV reconstructed using Clupatra and SiliconTracking_MarlinTrk
then combined into full tracks using FullLDCTracking_MarlinTrk

Introduction

Clupatra ...

- is a Pattern Recognition package designed for *ILD TPC* tracking;
- uses Kalman filter to fit tracks;
- was developed along with MarlinTrk by Frank Gaede and Steve Aplin to replace the old F77 tracking code;
- closely related to KalTest+KalDet(K. Fujii et. al.) / MarlinTrk;
- all packages are available in ILCSoft (most recent developments are to submitted soon);

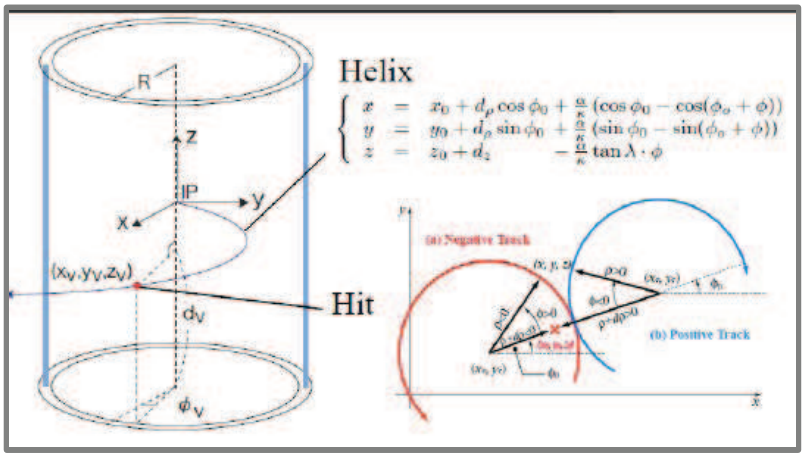
Introduction

Clupatra ...

- is a Pattern Recognition package designed for *ILD TPC* tracking;
- uses Kalman filter to fit tracks;
- was developed along with MarlinTrk by Frank Gaede and Steve Aplin to replace the old F77 tracking code;
- closely related to KalTest+KalDet(K. Fujii et. al.) / MarlinTrk;
- all packages are available in ILCSoft (most recent developments are to submitted soon);
- is now compatible with Test Beam data.

KalTest/KalDet Tools

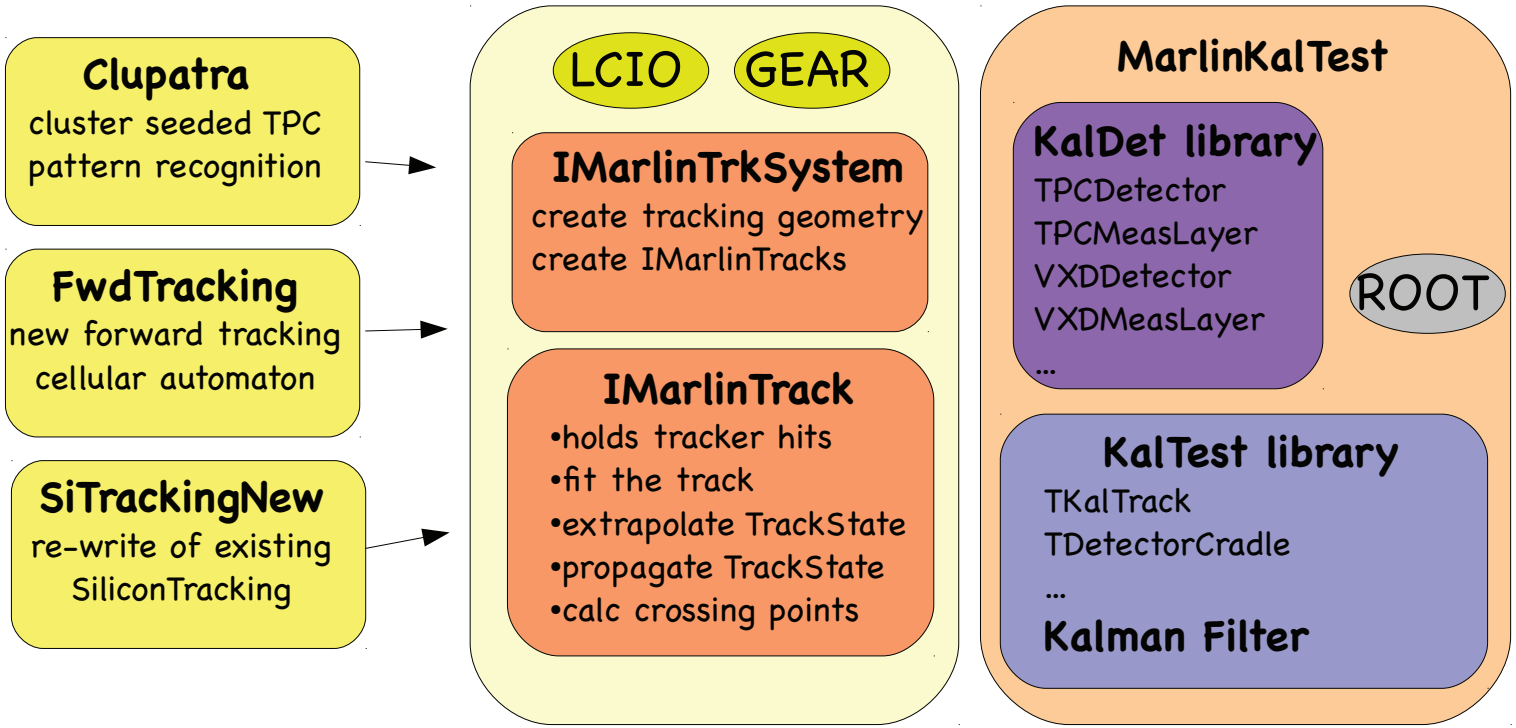
- KalTest
  - Kalman Filter tool (K. Fujii et al)
  - based on ROOT
    - TGeo, TMath, TObjArray
  - developed in Jupiter framework now included in iLCSoft
- KalDet
  - define detector geometry
    - meas. layer, coordinate to track state transform. ...
    - position of meas. layer and material properties
- both packages also used in MarlinTPC testbeam software



track parameters very close to canonical LCIO set:
d0, phi0, omega, tanL, z0
-> trivial transform

the MarlinTrk interface for tracking

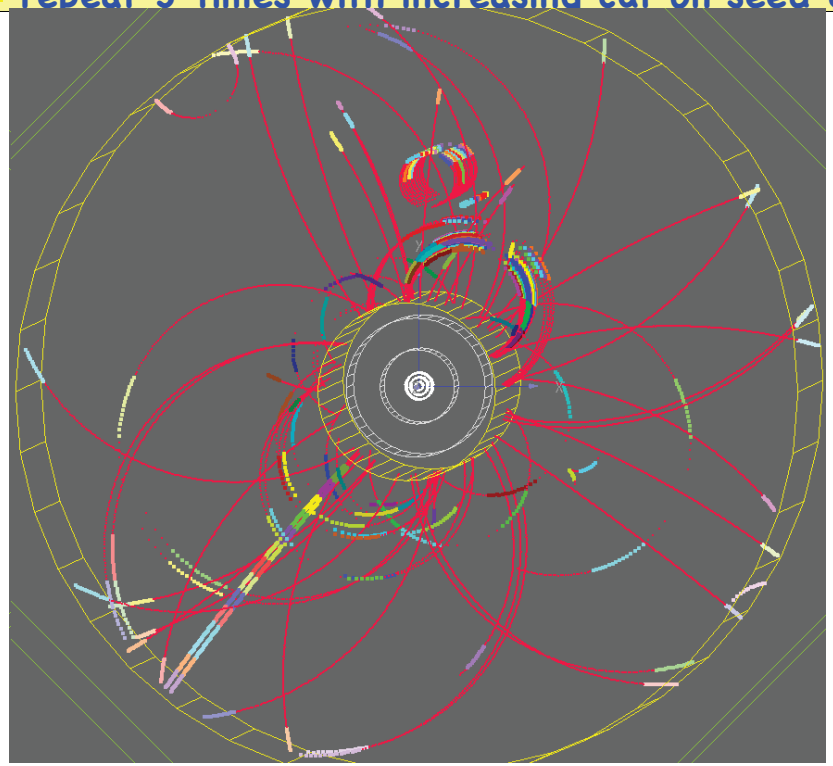
- new common API for developing tracking code (TPC, Silicon, Fwd)
- provides loose coupling between patrec and fitting
- defined abstract interface IMarlinTrk and implement using KalTest/KalDet
- currently lives in MarlinTrkProcessors



Pattern recognition algorithm in Clupatra

Clupatra step 1

- NN-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 !
  - search in next row
- repeat 3 times with increasing cut on seed clustering

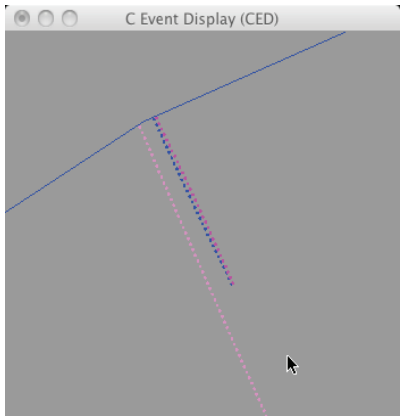


example:

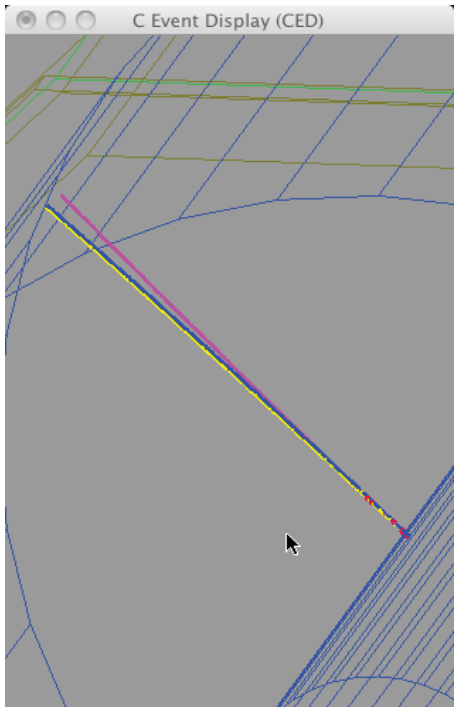
- ttbar event @ 500 GeV
- results in clean tracks and segments for curlers
- little leftover hits
- some very close by tracks lost (fixed in step2)

Clupatra step 2

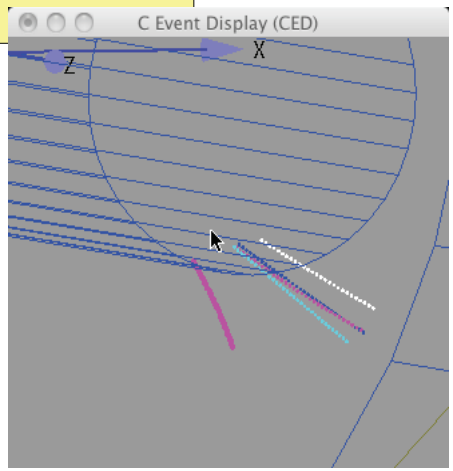
- re-cluster in leftover hits (NN clustering)
- based on pad row multiplicity force into
  N=2,....9 clusters
- apply KalTest fit to throw out falsely merged hits (rare)
  - higher multiplicity: repeat iteratively in smaller row ranges until only three or two tracks left



- 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 3

- repair split tracks:
  - **identify incomplete track segments** that:

    don't start at the inner field cage and/or that don't end at the outer field cage or endplate
  - **merge segments that have consistent tracks states** (based on delta chi2 after hits are added)
- problem mostly due to double hit resolution (merged hits)

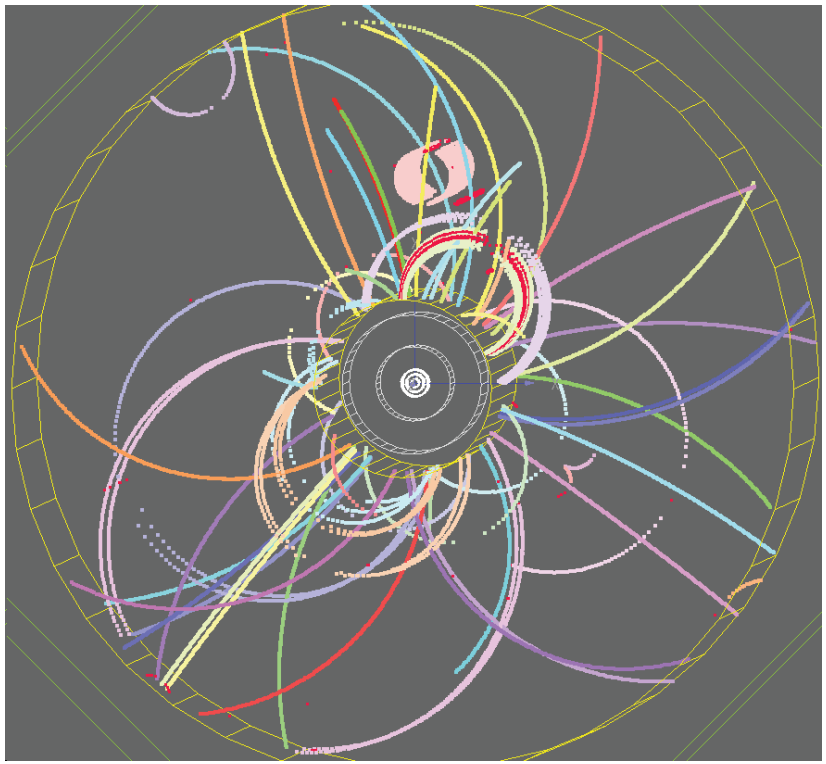
example: WW @ 1TeV
one lower pt track crossing four higher pt tracks in a dense jet

12

Frank Gaede, CLIC WS, CERN, Jan 28- Feb 01, 2013

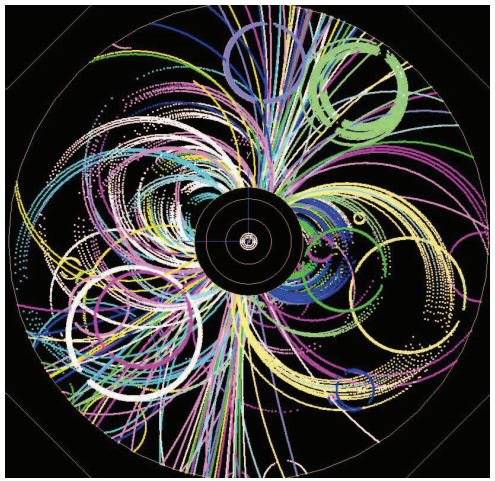
Clupatra step 4

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



examples:

- ttbar event @ 500 GeV
- only few segments are not merged
- most of these curler segments
- where lost in old patrec
- also works in higher multiplicities, e.g. @ 3 TeV:



13

Frank Gaede, CLIC WS, CERN, Jan 28- Feb 01, 2013

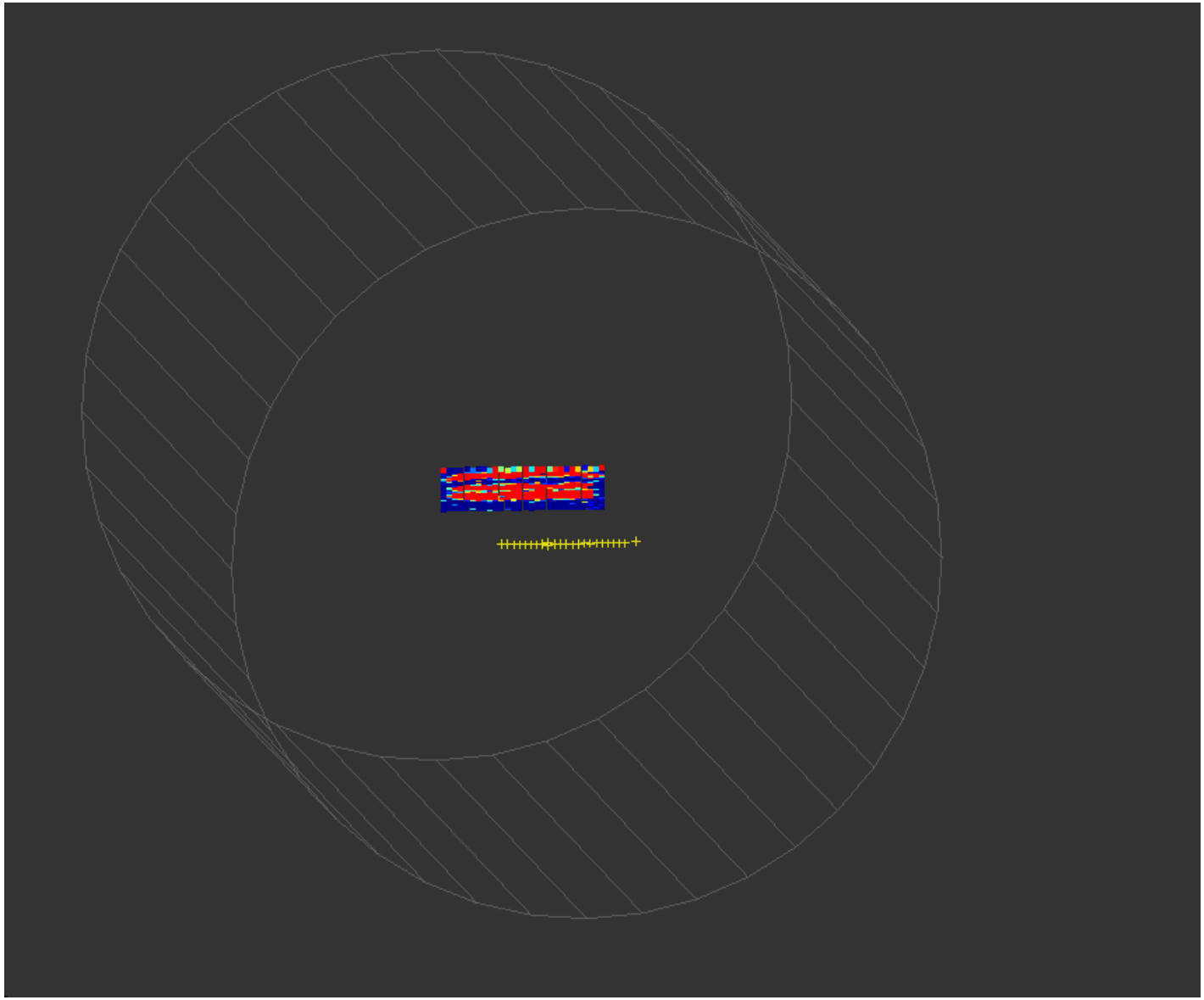
Changes in the packages related to Test Beam data

Slight modifications of the code due to different geometry treatment (see details in backup):

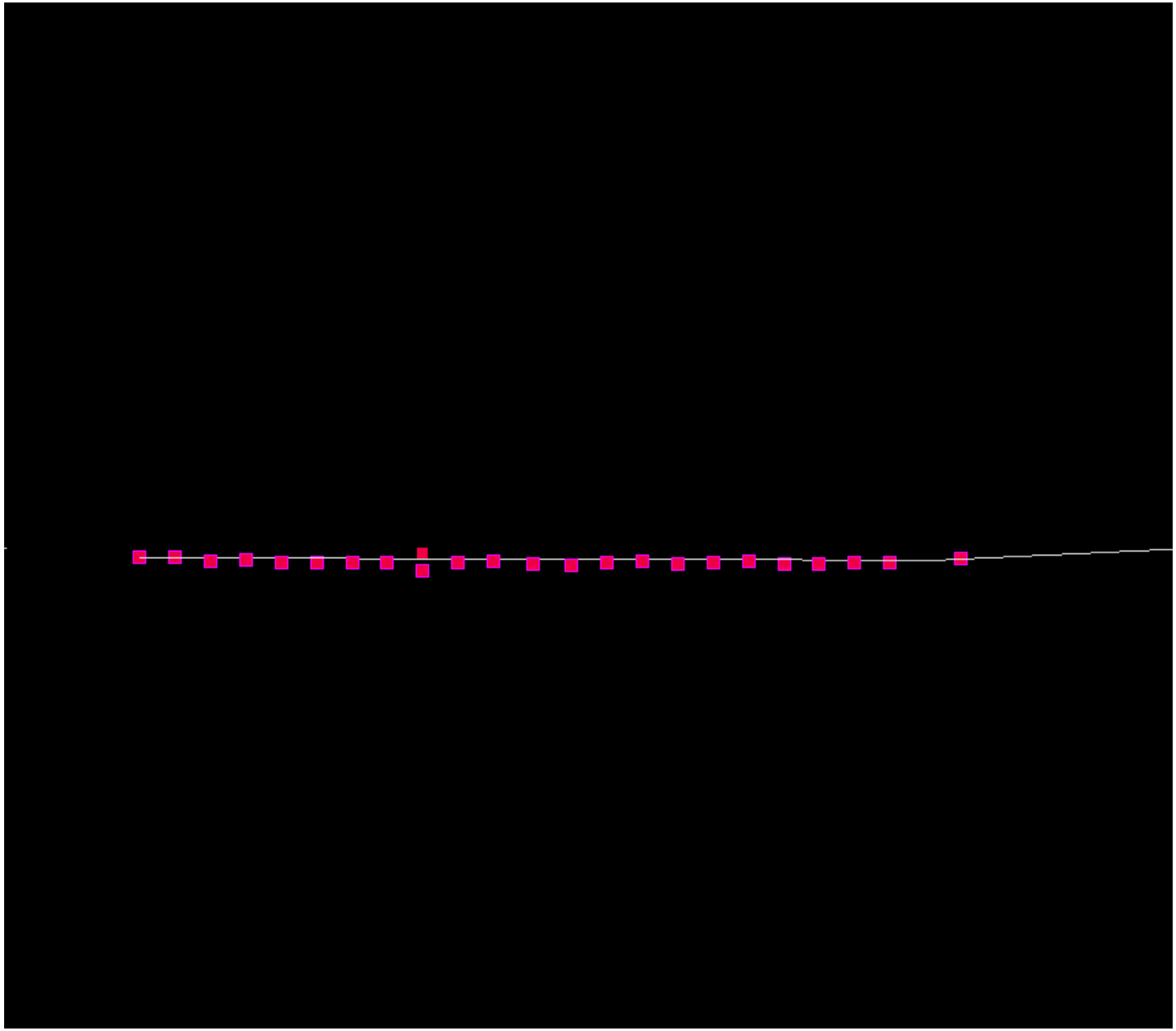
- KalDet: included the updated description of the LP (**ild/lctpc_test/LCTPCKalDetector**)
- Clupatra/KalDet: included the canonical ILD-type hit-**CellID** representation and corrected hit errors
- MarlinTrk: geometry treatment (use only TPC instead of all ILD detector parts, no track state AtCalorimeterFace etc.)
- MarlinUtil/MarlinCED, CEDViewer, MarlinTPC/TrackerRawViewer: visualization [in progress]

Fitting results

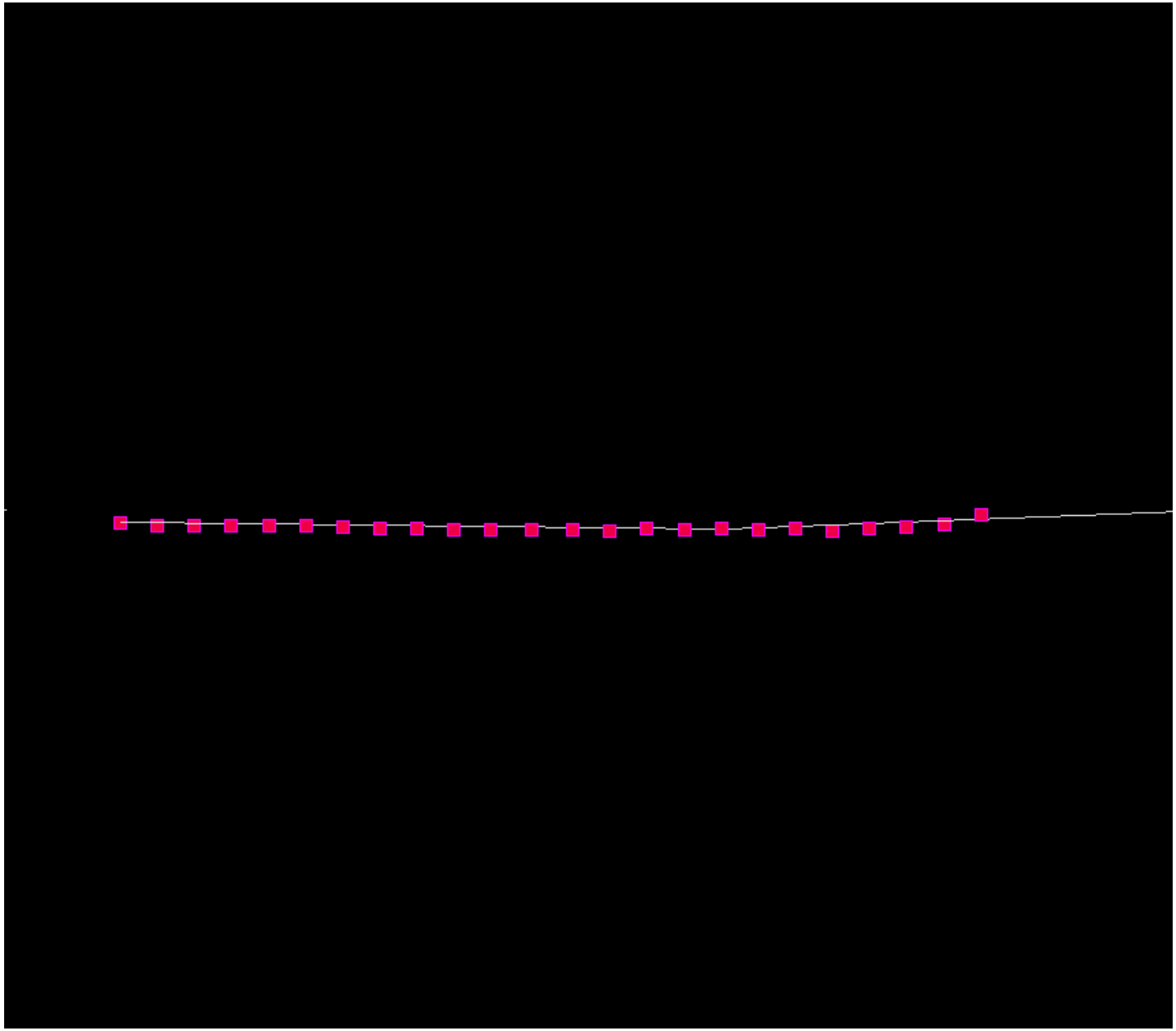
1 module data geometry sketch



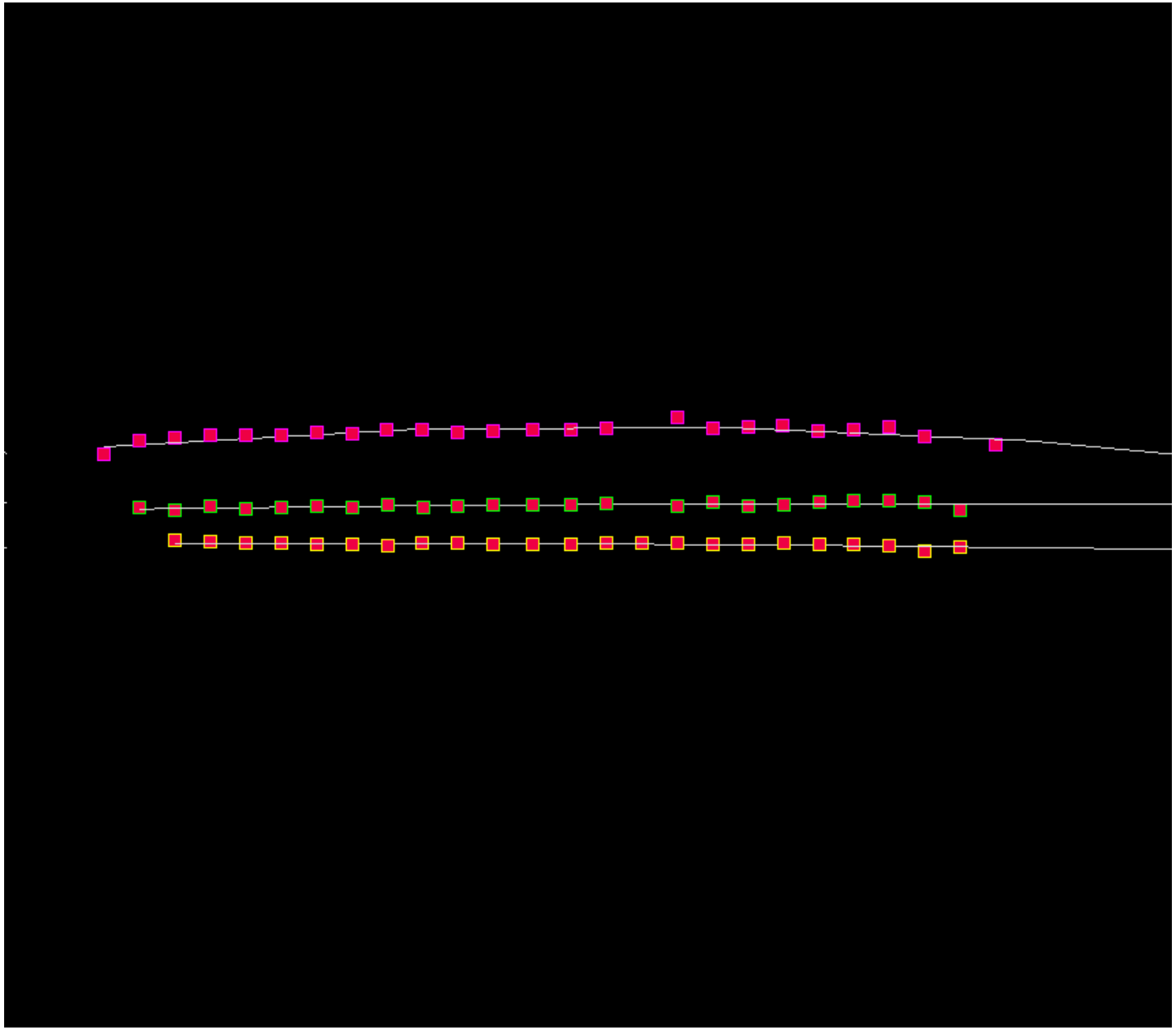
1-module data (June 2011), run 17637. Most of the fits look very good:



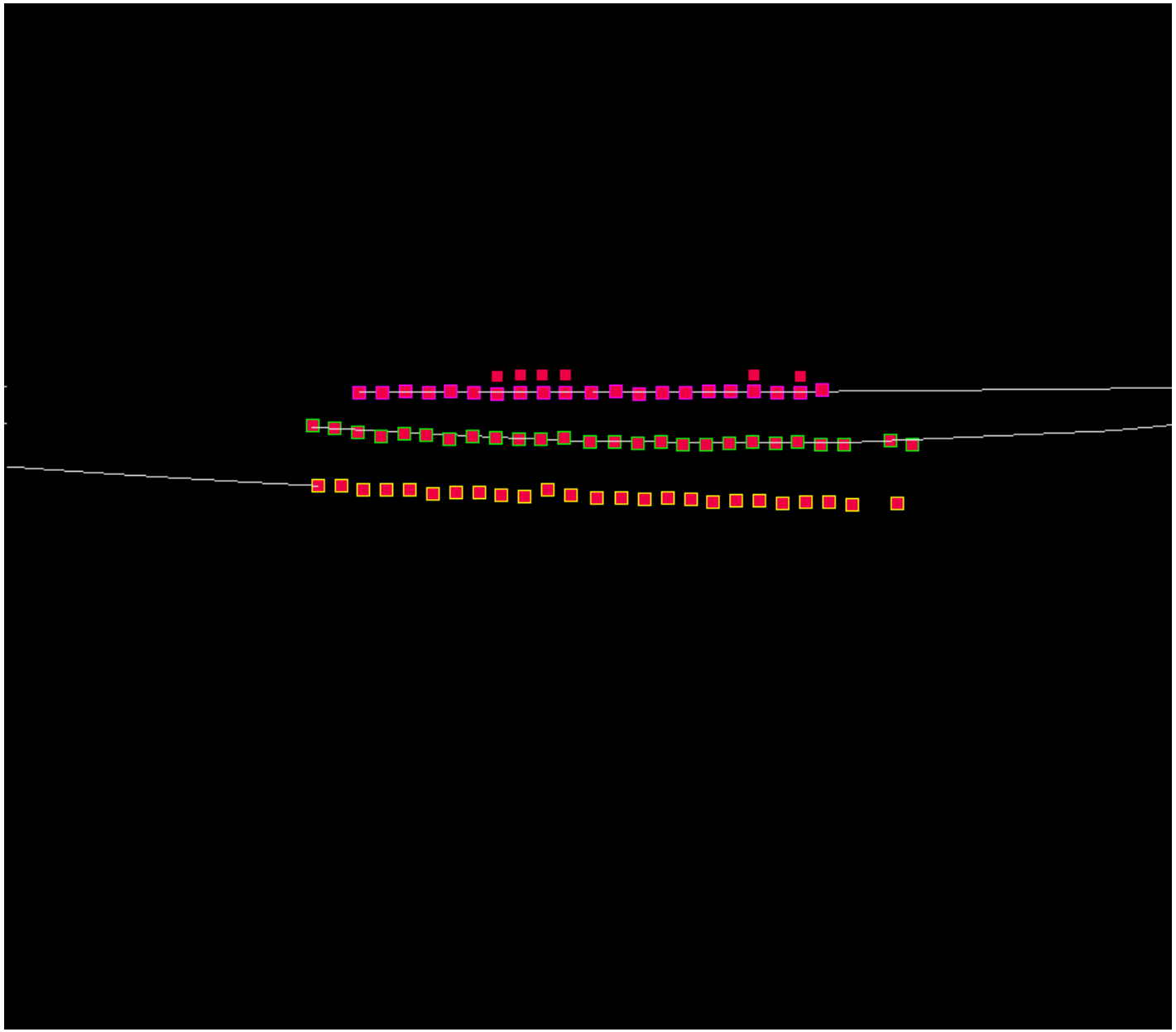
1-module data (June 2011), run 17637. Most of the fits look very good:



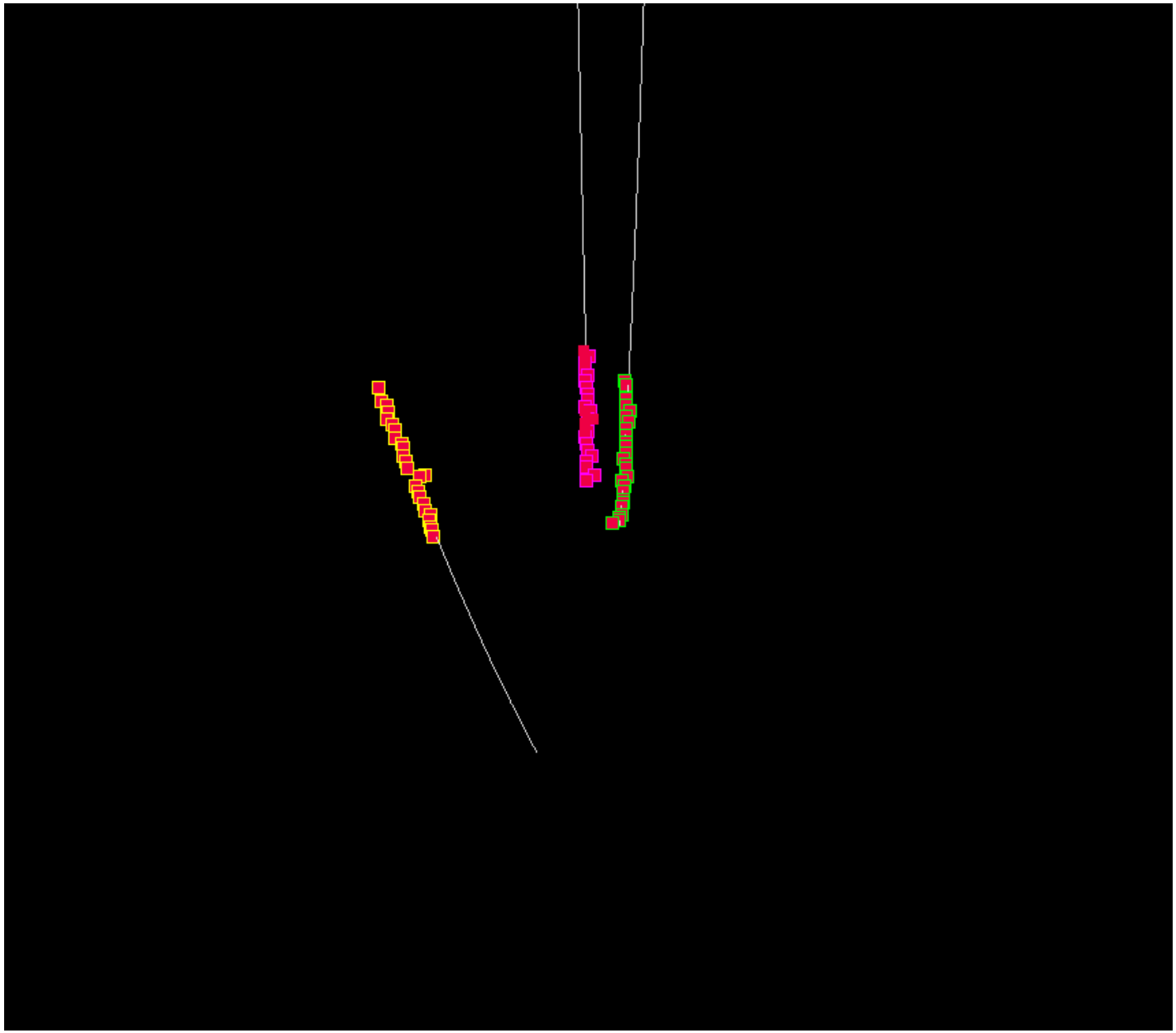
1-module data (June 2011), run 17637. Most of the fits look very good:



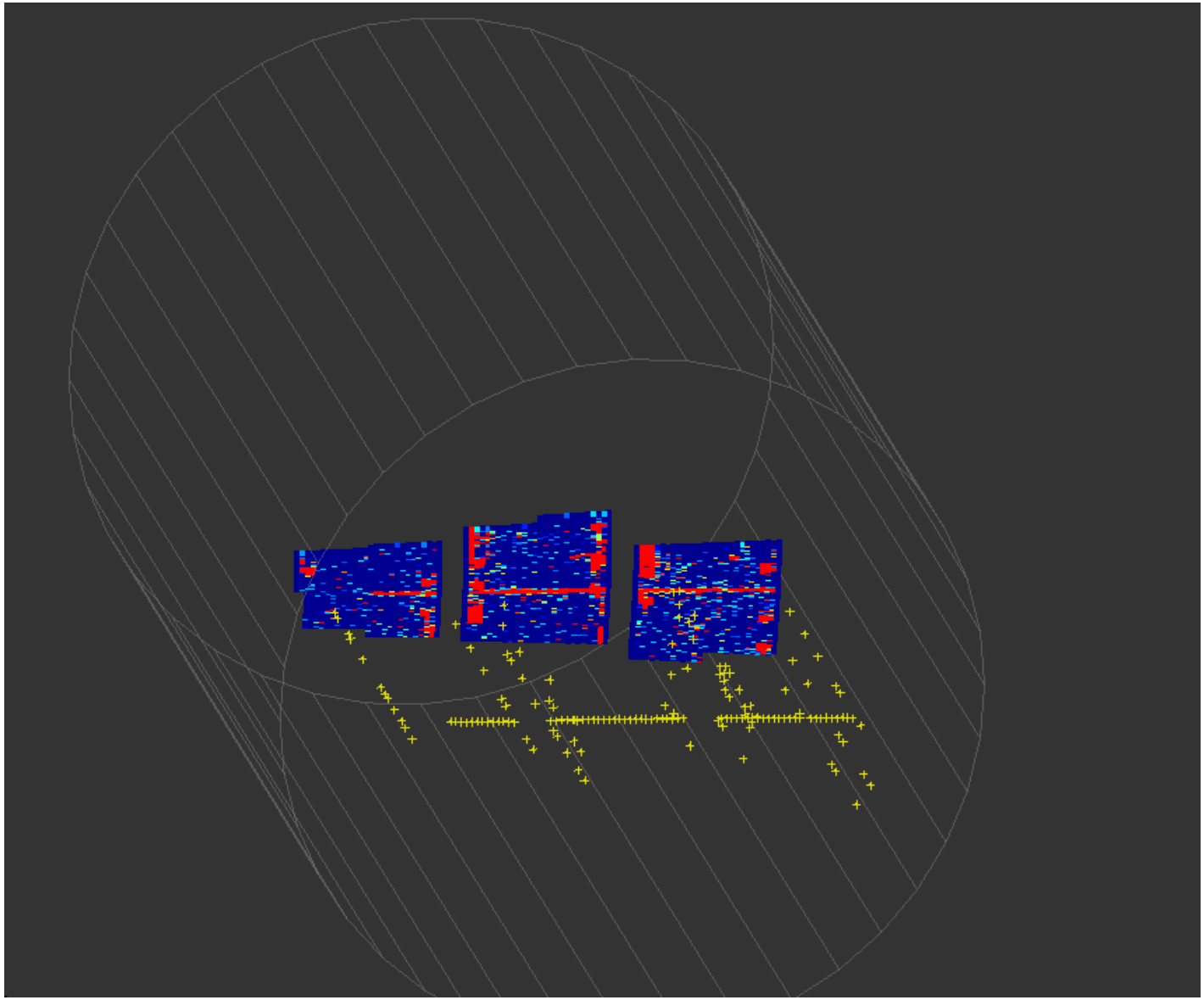
1-module data (June 2011), run 17637. 1 track has wrong direction (the IP is assumed to be far on the left ($z = 0$):



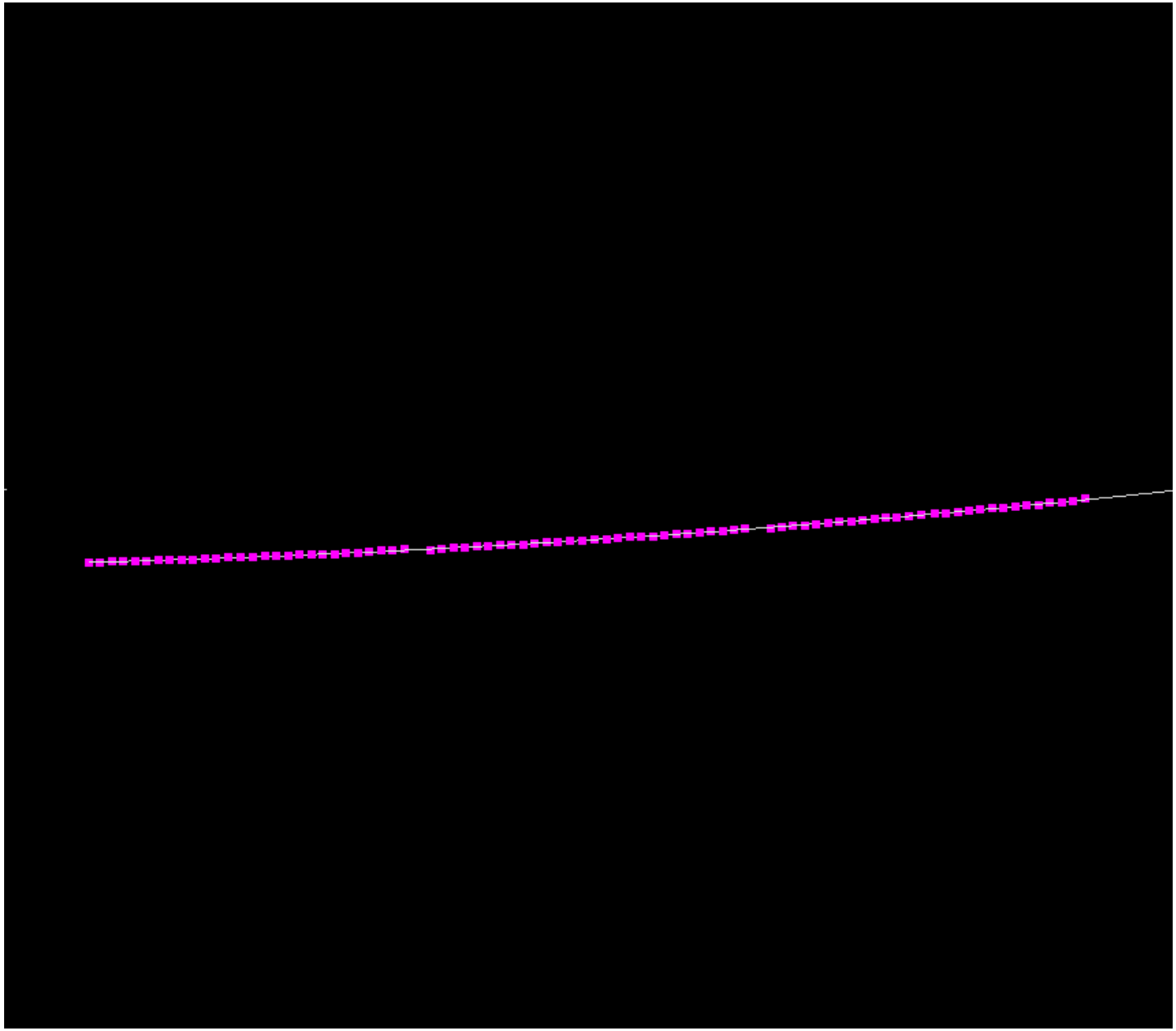
1-module data (June 2011), run 17637. 1 track has wrong direction (the IP is assumed far on the left):



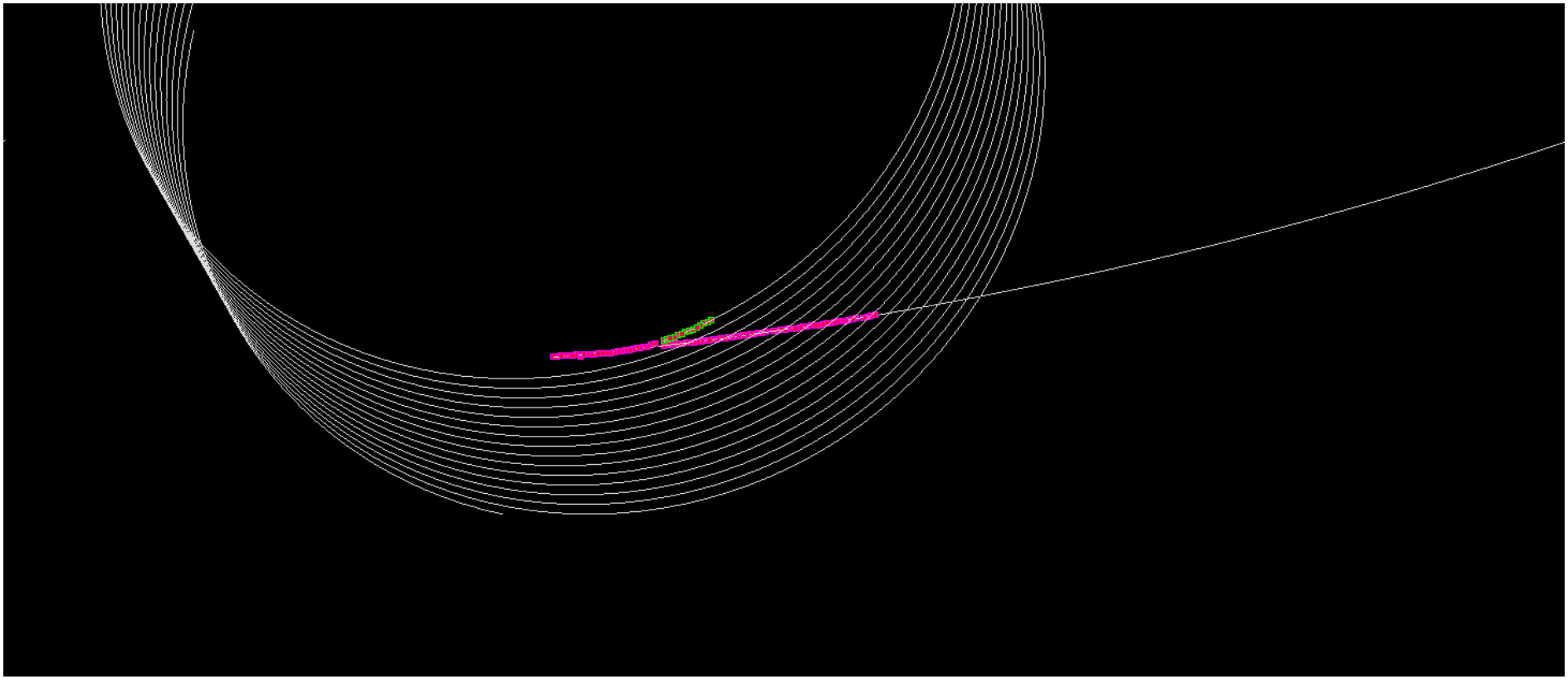
3 modules geometry sketch



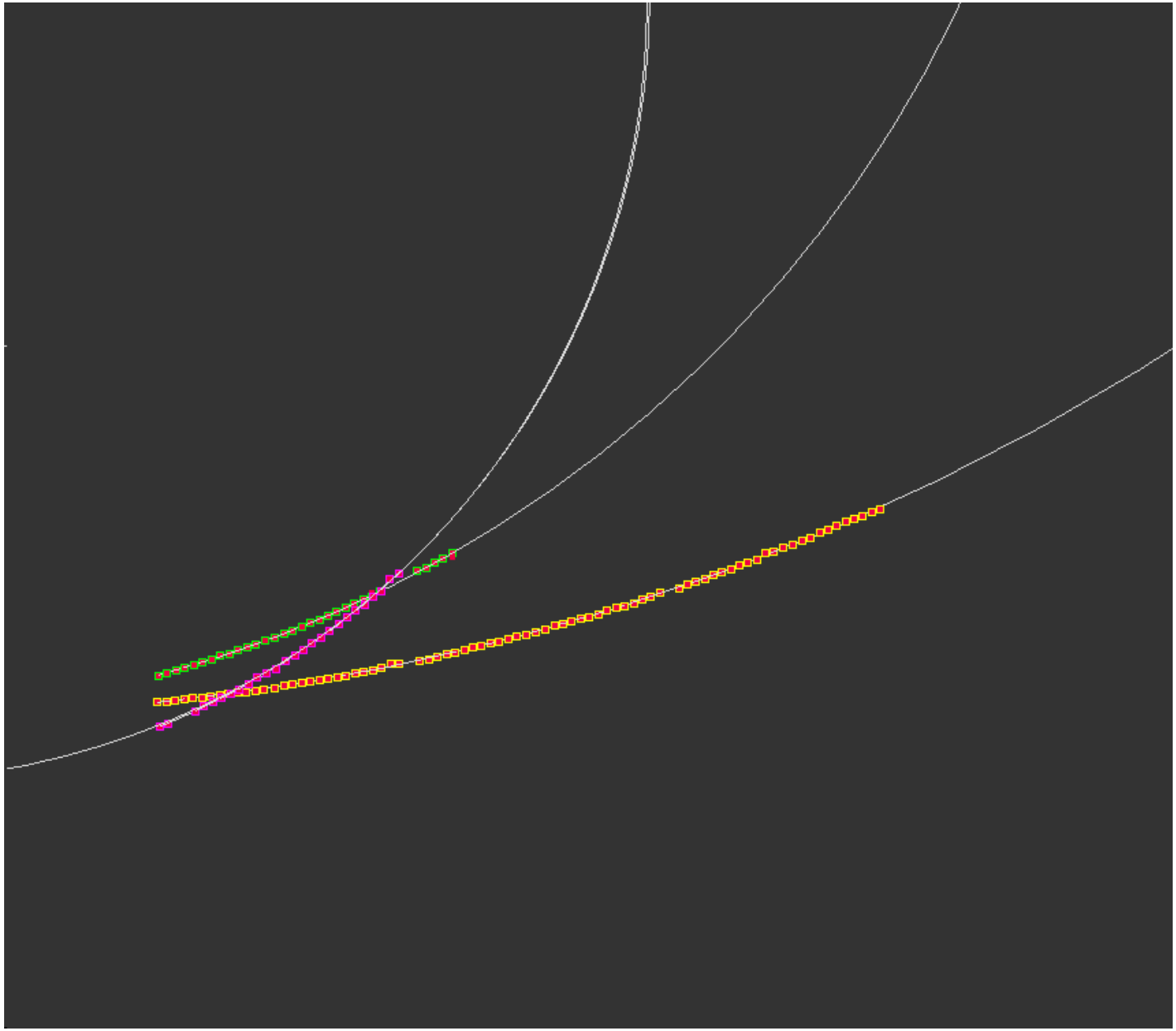
3-module data (Mar 2013):



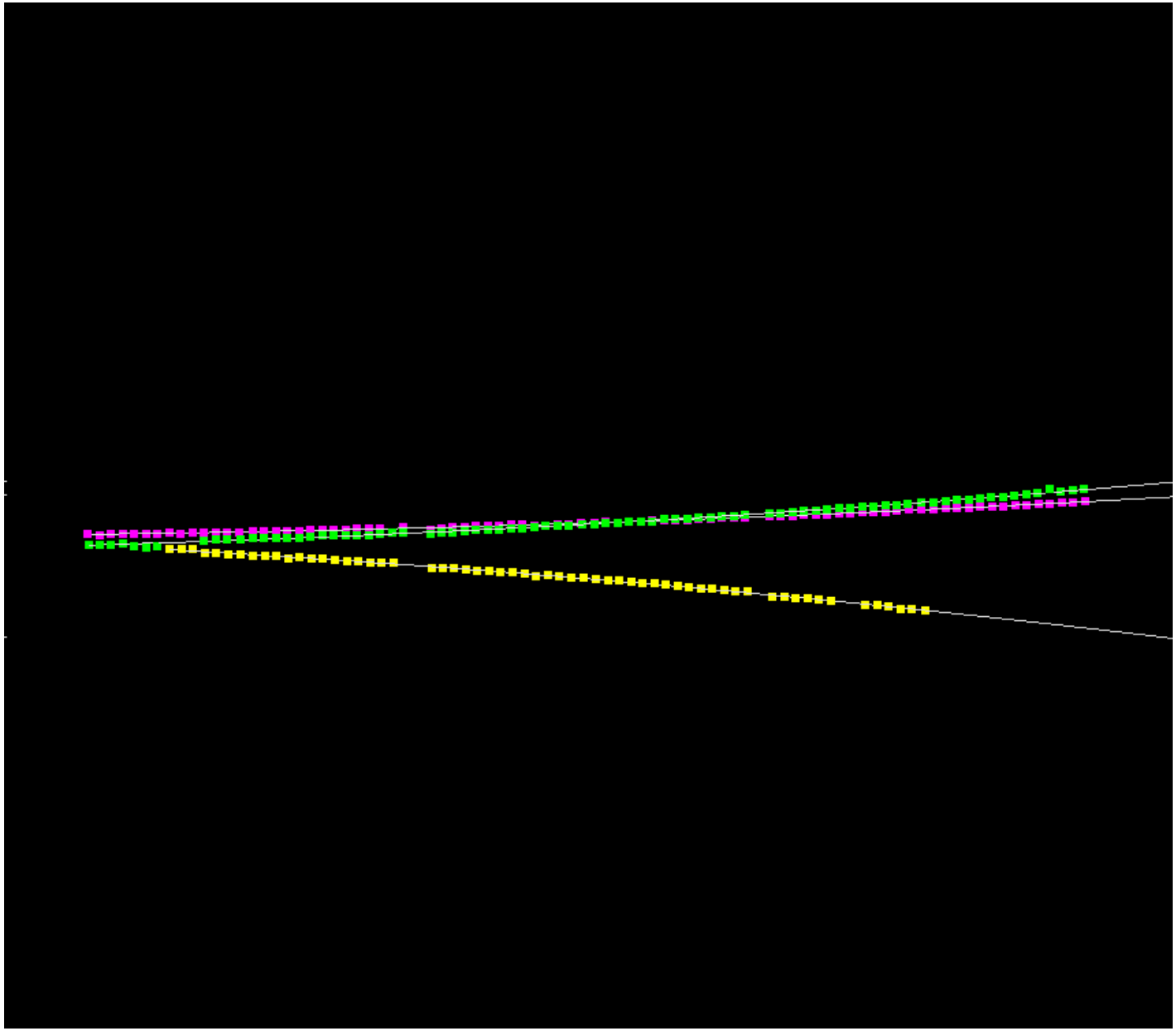
3-module data (Mar 2013):



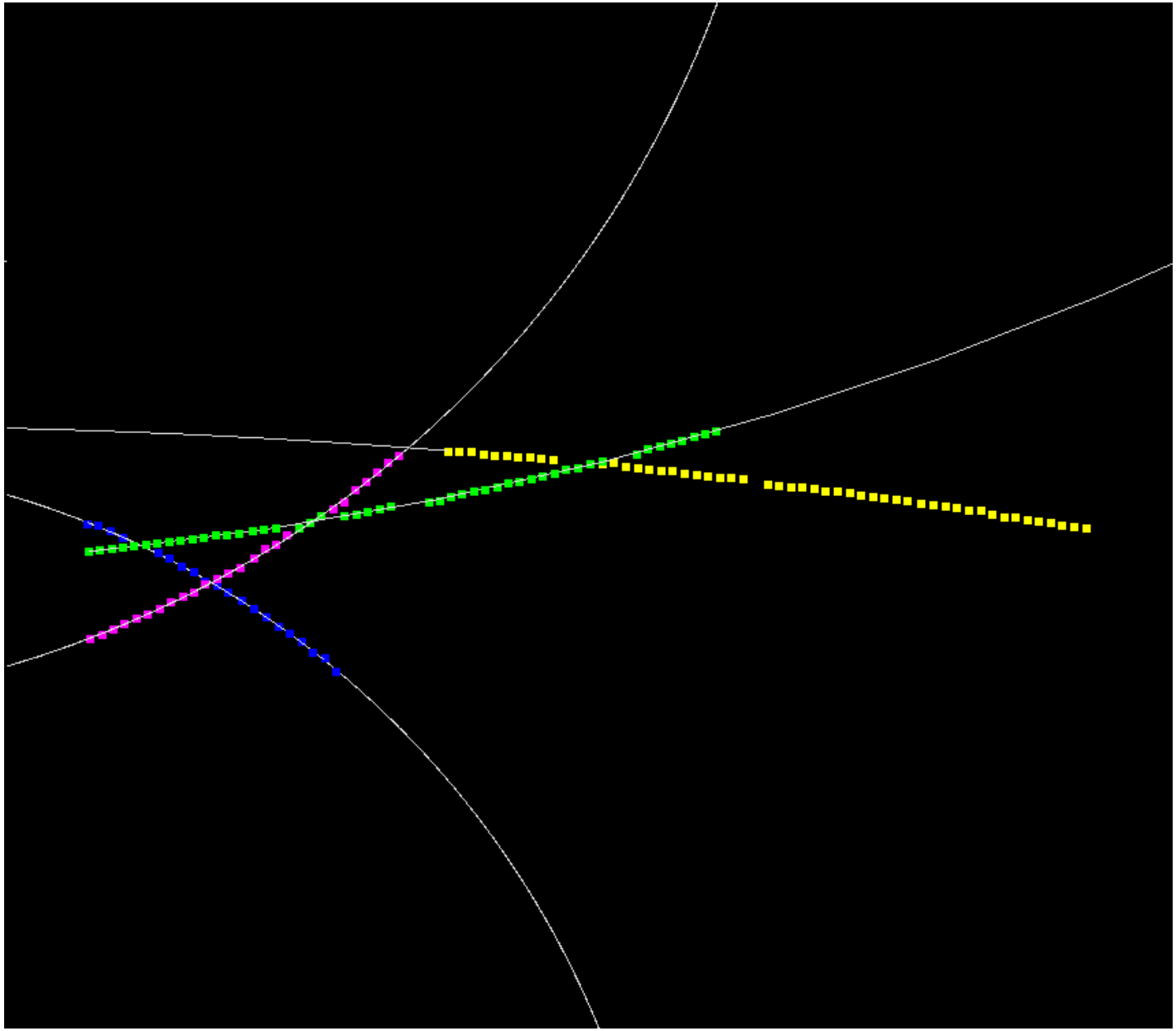
3-module data (Mar 2013):



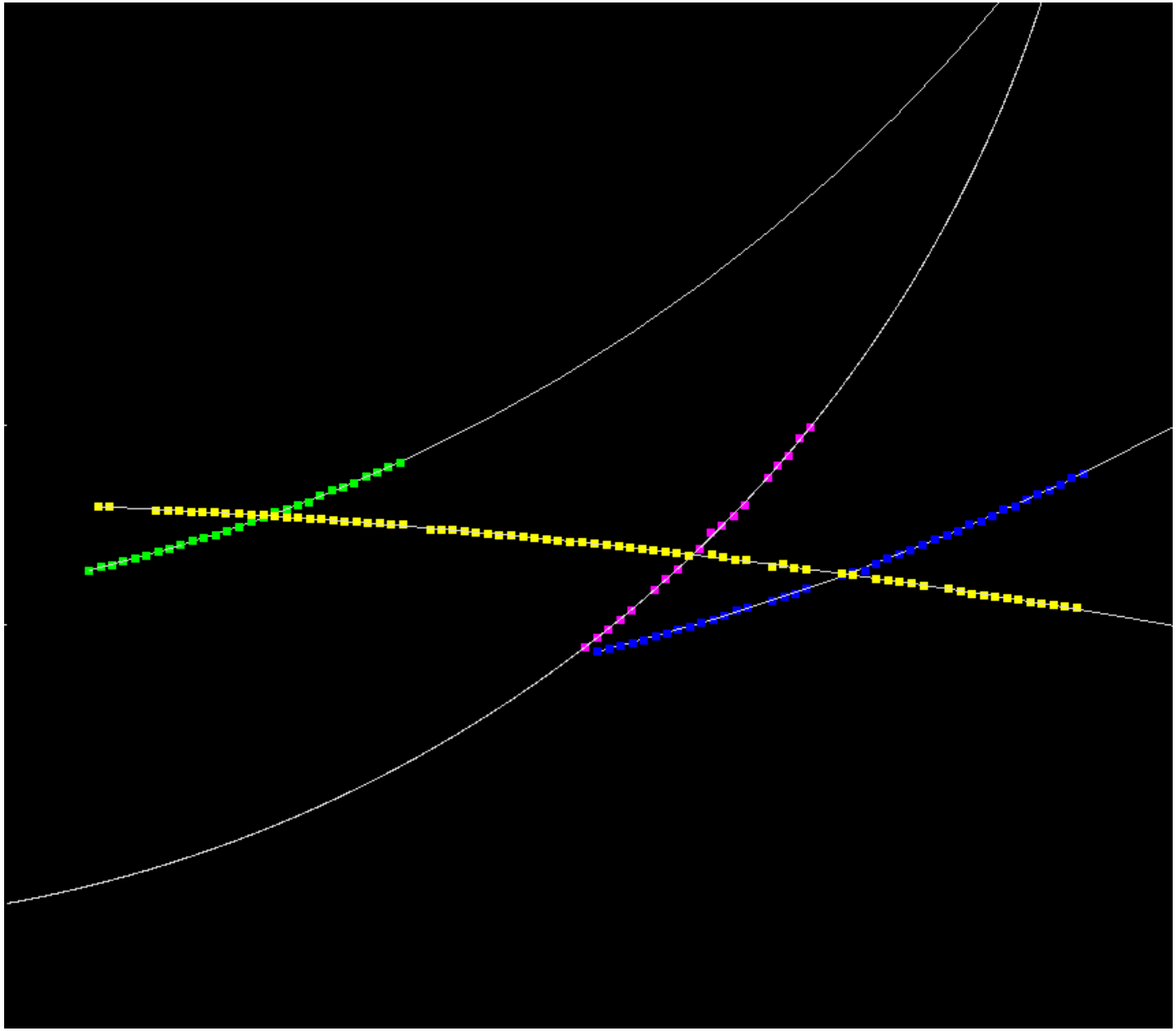
3-module data (Mar 2013):



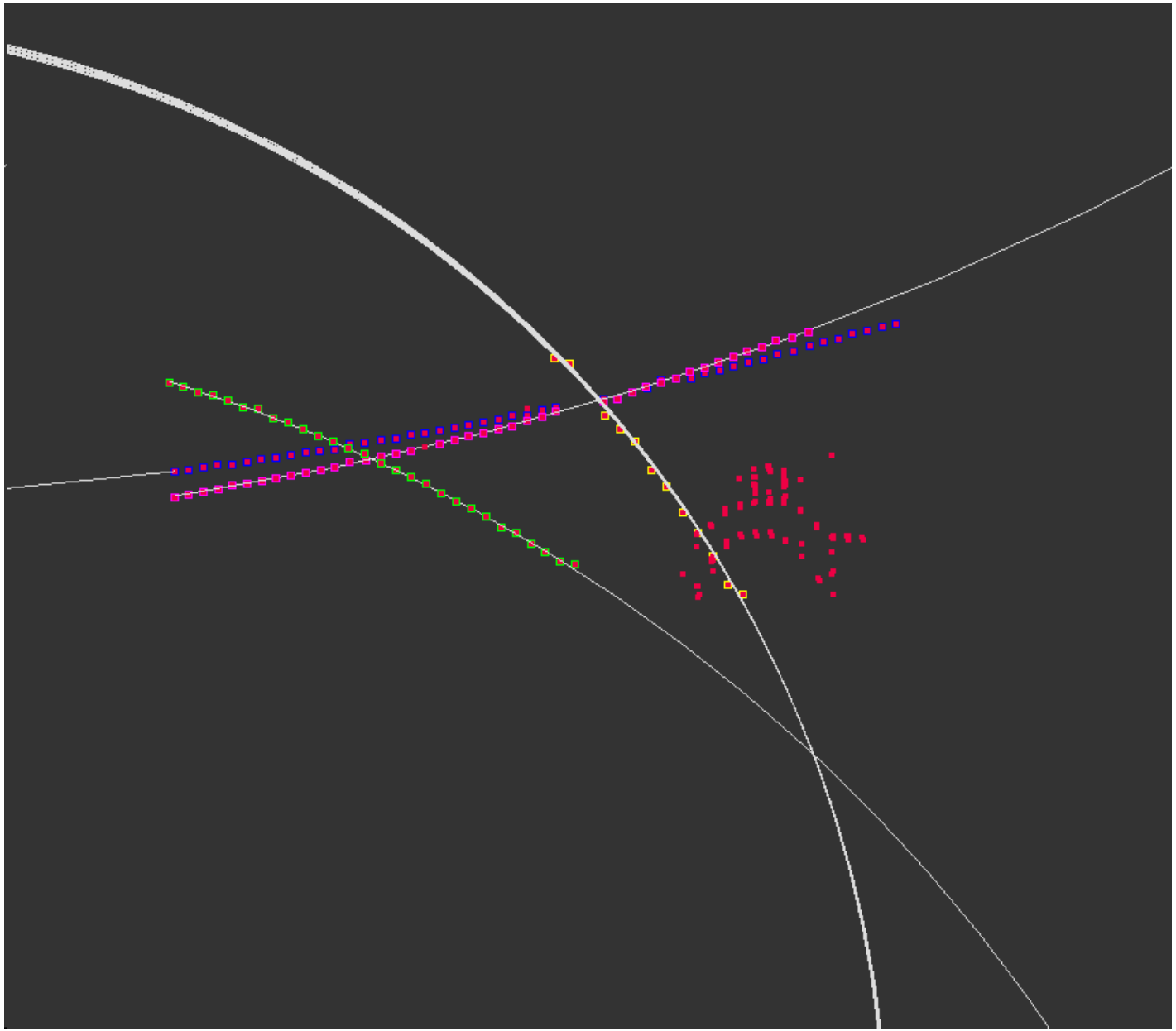
3-module data (Mar 2013):



3-module data (Mar 2013):



3 modules. Crowded event, but still reasonable results:



Benchmarking

Data type	Clupatra	Pathfinder (helix track model)
3 Module (B field ON)	0.04 s/evt	1.6 s/evt
1 Module (B field OFF)	0.009 s/evt	0.16 s/evt

Clupatra parameters

- Fit results are strongly dependent on the data (data quality, noisiness etc.).
- No versatile Clupatra processor settings
- Wrong settings may results in non-satisfactory results

Clupatra parameters

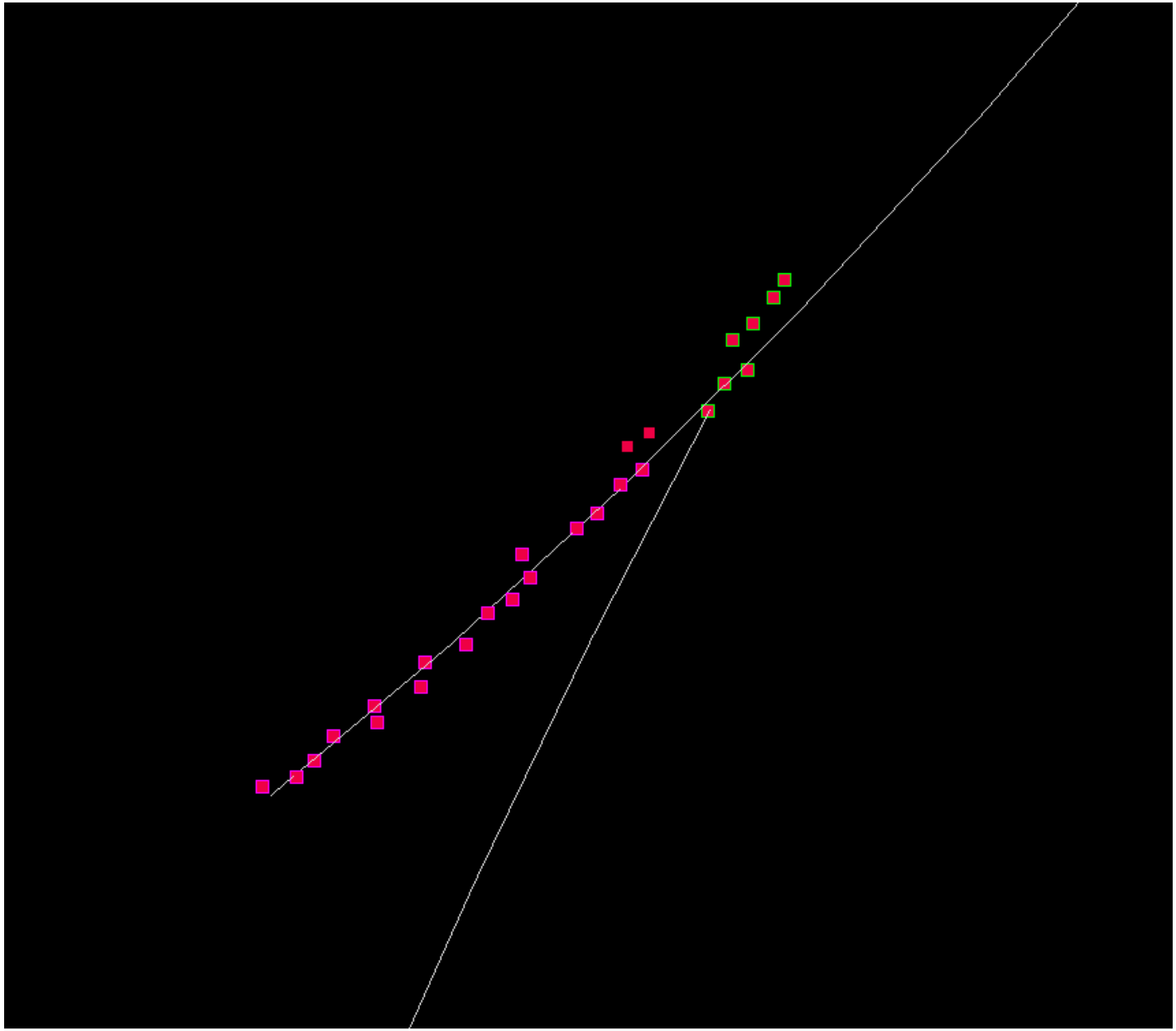
- Fit results are strongly dependent on the data (data quality, noisiness etc.).
- No versatile Clupatra processor settings
- Wrong settings may results in non-satisfactory results

check the Clupatra input parameters e.g. using CED viewer

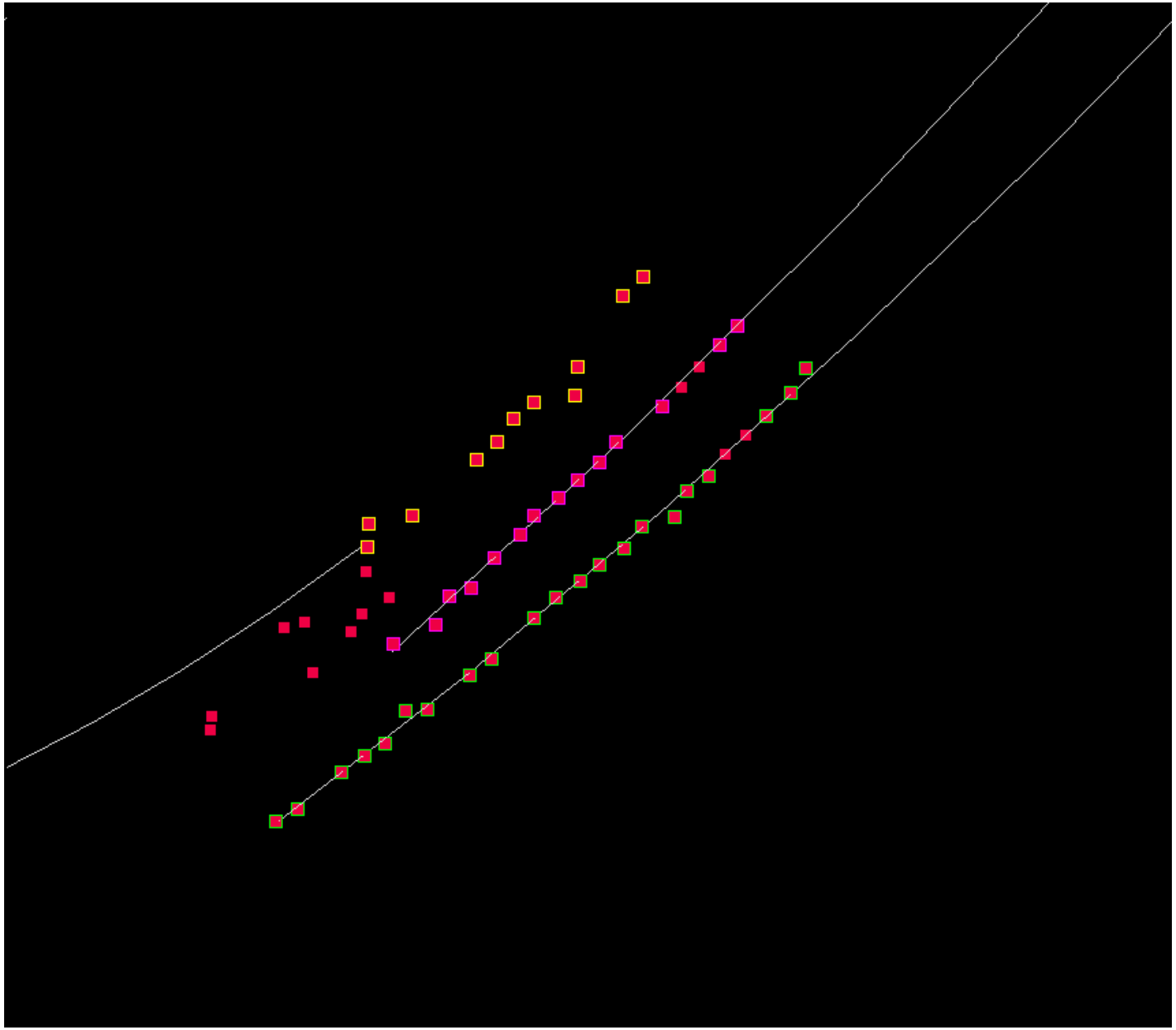
Important Clupatra parameters

```
<!--the maximum chi2-distance for which a hit is considered for merging -->
<parameter name="Chi2Cut" type="float">40 </parameter>

<!--the maximum delta Chi2 for which a hit is added to a track segement -->
<parameter name="MaxDeltaChi2" type="float">40 </parameter>

<!--Cut for distance between hits in mm -->
<parameter name="DistanceCut" type="float">40 </parameter>

<!--the maximum number of layers without finding a hit before hit search is stopped -->
<parameter name="MaxStepWithoutHit" type="int">5 </parameter>

<!--minimum number of hits per cluster -->
<parameter name="MinimumClusterSize" type="int">5 </parameter>

<!--number of pad rows used in initial seed clustering-->
<parameter name="PadRowRange" type="int">5 </parameter>
```

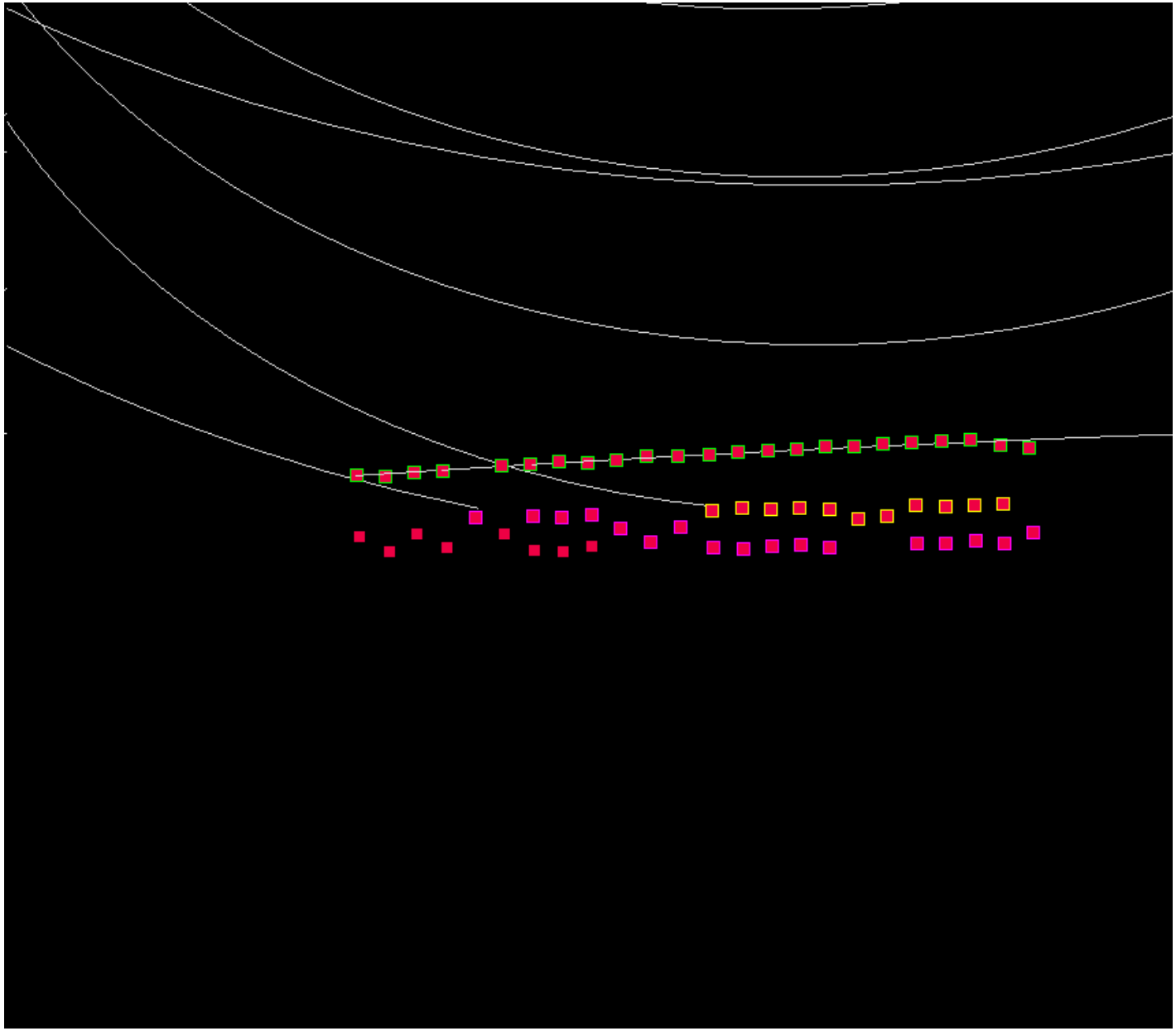
1 module. Noisy event / too small χ^2 for adding hits to the fit:



1 module. Noisy event / too small χ^2 for adding hits to the fit:



1-module data (June 2011), run 17637. Crowded event:



Clupatra+TB data known issues

- Straight tracks not yet supported. For the moment you need to set $B_z \neq 0$ (e.g. $B_z = 1.0$) in the gearfile
- Track direction not yet clearly defined for the TB data (tracks in ILD originate from the IP and thus their directions depend on the hit order, charge, Ω_{track} etc.)
- Some visualization issues - in progress...

Summary

- Clupatra has been adopted for the TB data track fitting - unified ILD + TB tracking is now available
- The fit may be sensitive to the type of data used - needs adjustment of the Clupatra processor parameters
- Some minor issues remain but they do not affect the track fit results

References

1. S. Aplin @ LCWS (Linear Collider Workshop) 2011, Granada, Spain, Sep 28, 2011
2. S. Aplin @ ILD Software and Integration Meeting, Nov. 30, 2011
3. F. Gaede @ CLIC Workshop, CERN, Jan 28 – Feb 01, 2013

Backup

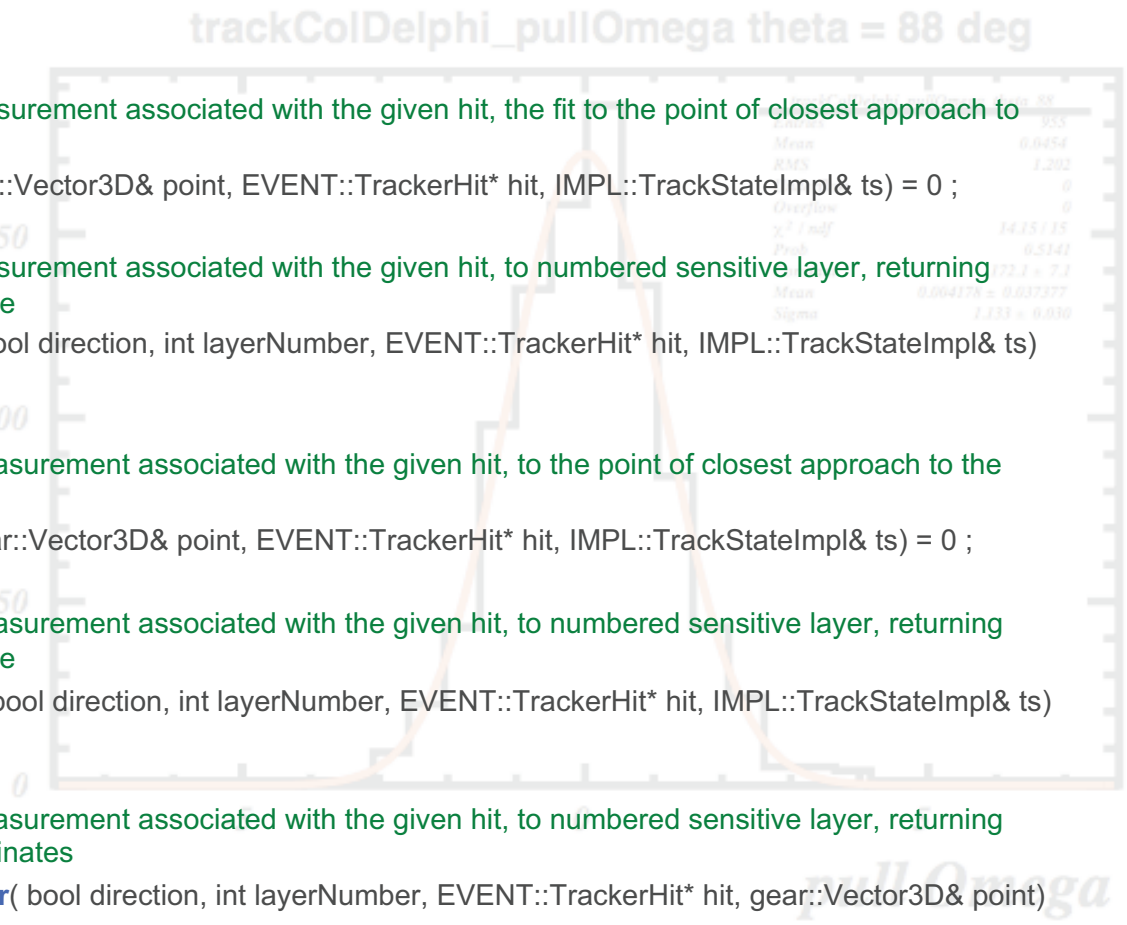
IMarlinTrack and IMarlinTrkSystem

- **IMarlinTrack** interface should provide a convenient interface when using an iterative fitter and also during pattern recognition.
- Examples of methods provided:

```
//** initialise the fit using the supplied hits only, using the given order to determine the direction of the track
virtual int initialise( bool direction ) = 0 ;

//** initialise the fit with a track state
virtual int initialise( const IMPL::TrackStateImpl& ts) = 0 ;

//** update the current fit using the supplied hit, return code via int. Provides the Chi2 increment to the fit from adding the hit via reference.
virtual int addAndFit( EVENT::TrackerHit* hit, double& chi2increment, double maxChi2Increment=DBL_MAX ) = 0 ;

//** get track state, return code via int
virtual int getTrackState( IMPL::TrackStateImpl& ts ) = 0 ;

//** get track state at measurement associated with the given hit, return code via int
virtual int getTrackState( EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts ) = 0 ;
```

continued …

IMarlinTrack and IMarlinTrkSystem



//** propagate track state at measurement associated with the given hit, the fit to the point of closest approach to the given point.

virtual int **propagate**(const gear::Vector3D& point, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** propagate track state at measurement associated with the given hit, to numbered sensitive layer, returning TrackState via provided reference

virtual int **propagateToLayer**(bool direction, int layerNumber, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to the point of closest approach to the given point.

virtual int **extrapolate**(const gear::Vector3D& point, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to numbered sensitive layer, returning TrackState via provided reference

virtual int **extrapolateToLayer**(bool direction, int layerNumber, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to numbered sensitive layer, returning intersection point in global coordinates

virtual int **intersectionWithLayer**(bool direction, int layerNumber, EVENT::TrackerHit* hit, gear::Vector3D& point) = 0 ;

Changes in KalDet

New class **ild/lctpc_test/LCTPCKalDetector**.

- based on existing **lctpc/GearTPCKalDetector** class;
- included the canonical ILD hit **CellID** representation;
- row numbering is "global": hits having the same local r coordinates belong to the same module, i.e. same measurement layer: rows of modules 0-1; 2-3-4; 5-6 are unified

Changes in MarlinTrk

Completed:

- included the use of the **LCTPCKalDetector** class
- minor changes related to the difference of the representation ILD-LCTPC (also see SVN logs):
  - measurement layers and TrackState-s at InteractionPoint/Calorimeter does not exist in LCTPC

TODO/In progress:

- include straight line representation for the fitting

Changes in Clupatra

- added **FixCellIDs_Errors_TestBeam** processor. Sets the hit CellID0 to canonical ILD values and hit uncertainties corresponding to the charge drift spread (based on the z coordinate)
- included multi-module support