Data Concentrator

Concepts, Implementation and Integration of the FPGA-based Tracking Algorithm into BASF2

Jochen Dingfelder, Carlos Mariñas,
Michael Schnell
schnell@physik.uni-bonn.de

University of Bonn

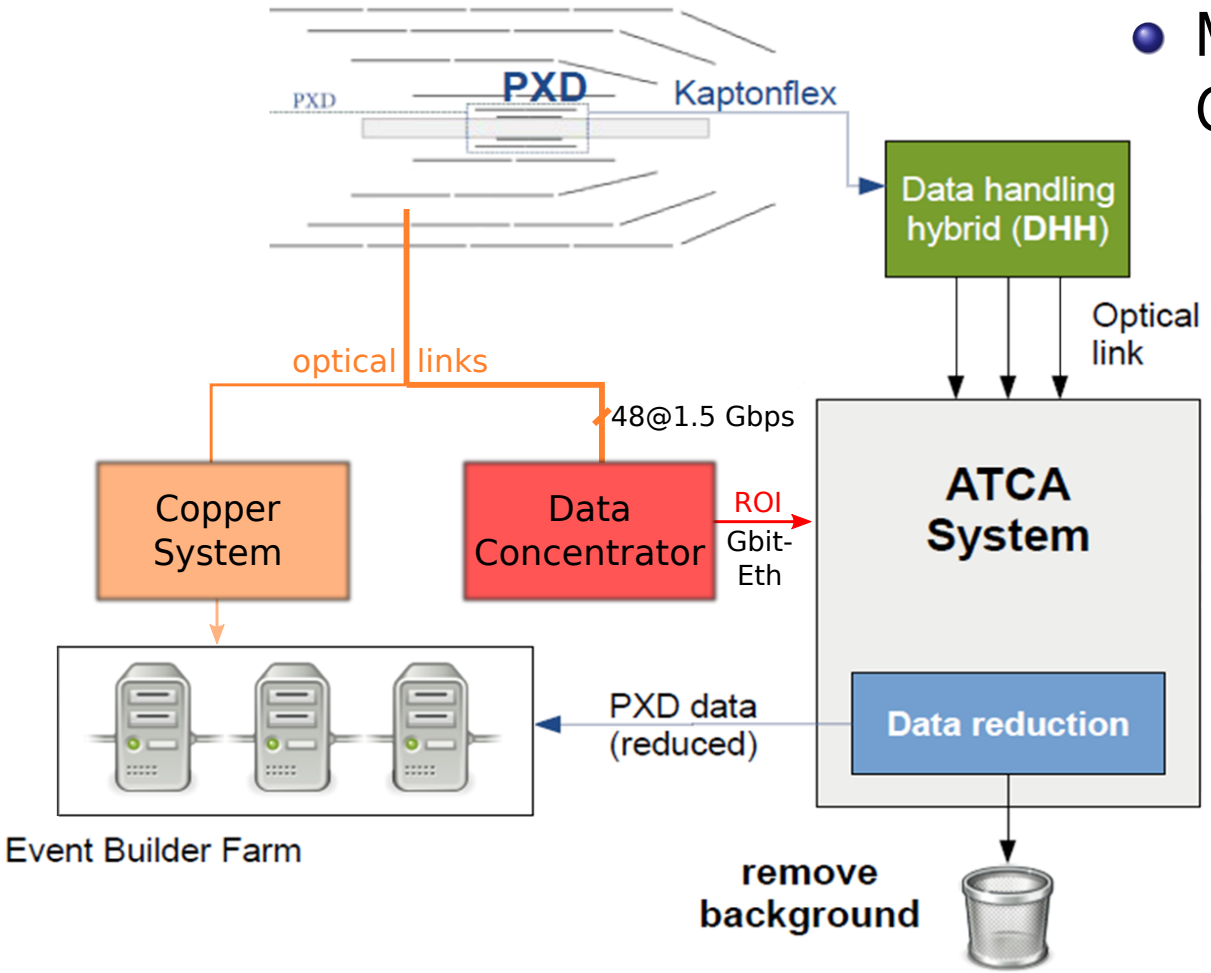
December 12th, 2013

Bundesministerium
für Bildung
und Forschung

universitätbonn

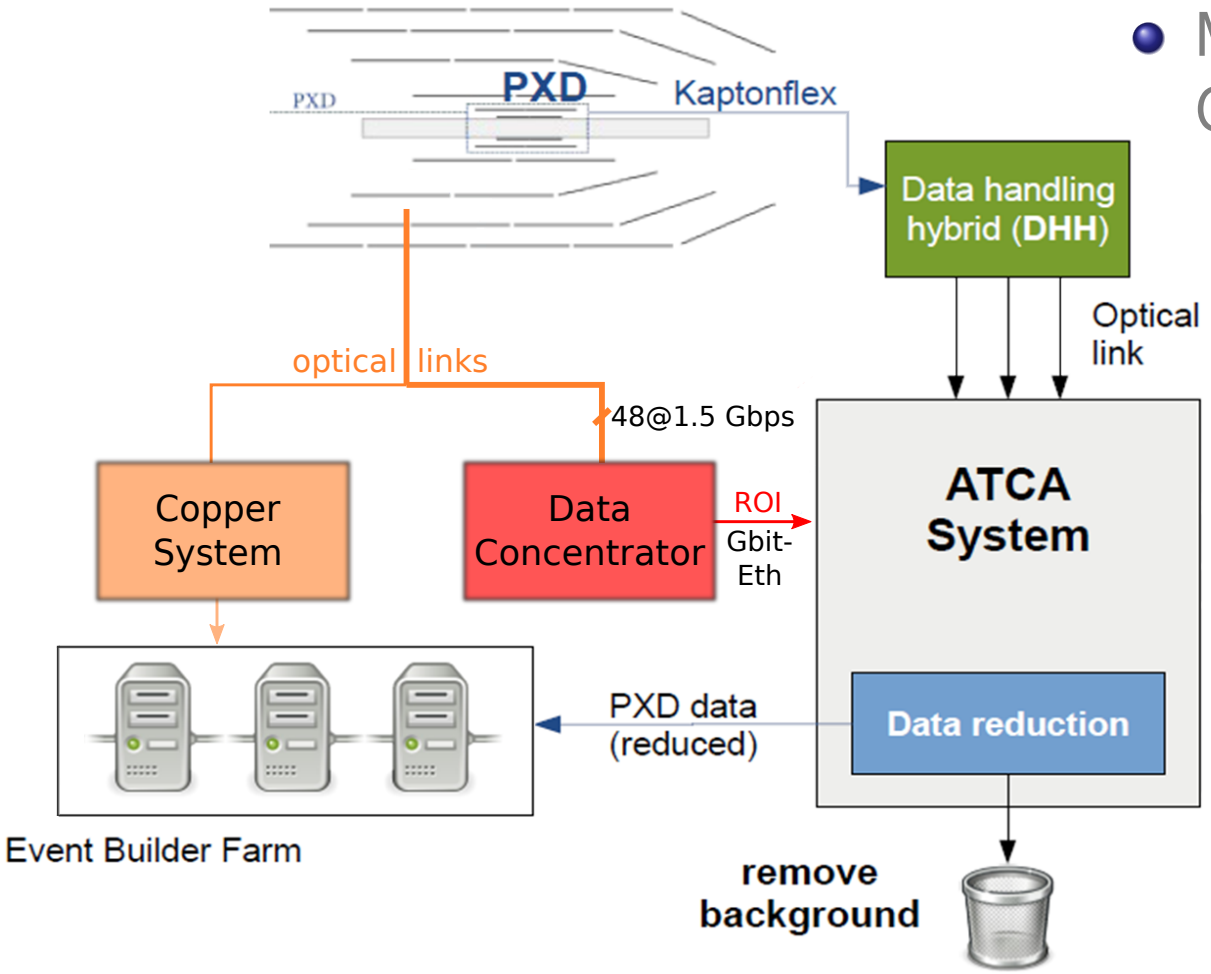
Content

Tasks of the Data Concentrator



- Major tasks of the Data Concentrator (DATCON):
  1. Acquire the data from the SVD on 48 optical links
  2. Reconstruct the track segments, extrapolation to PXD, ROI creation and broadcast to ATCA system over Gbit Ethernet

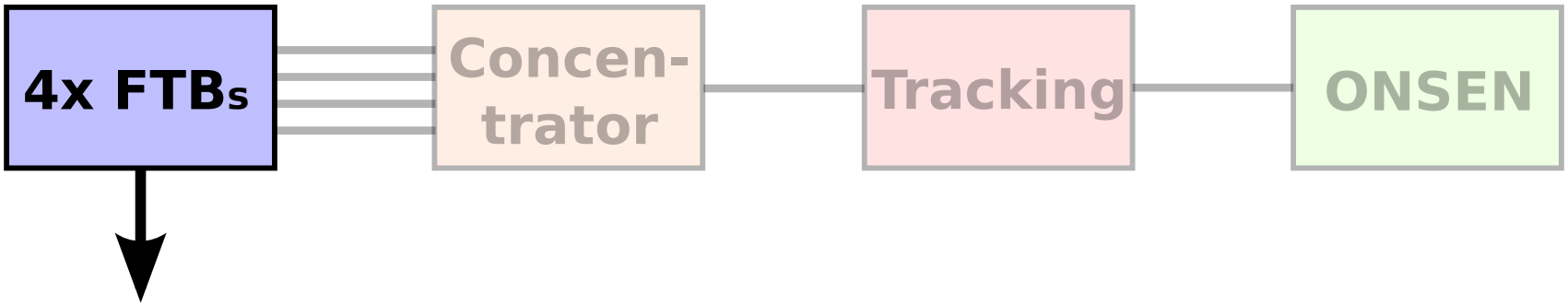
Tasks of the Data Concentrator



- Major tasks of the Data Concentrator (DATCON):
  1. Acquire the data from the SVD on 48 optical links
  2. Reconstruct the track segments, extrapolation to PXD, ROI creation and broadcast to ATCA system over Gbit Ethernet

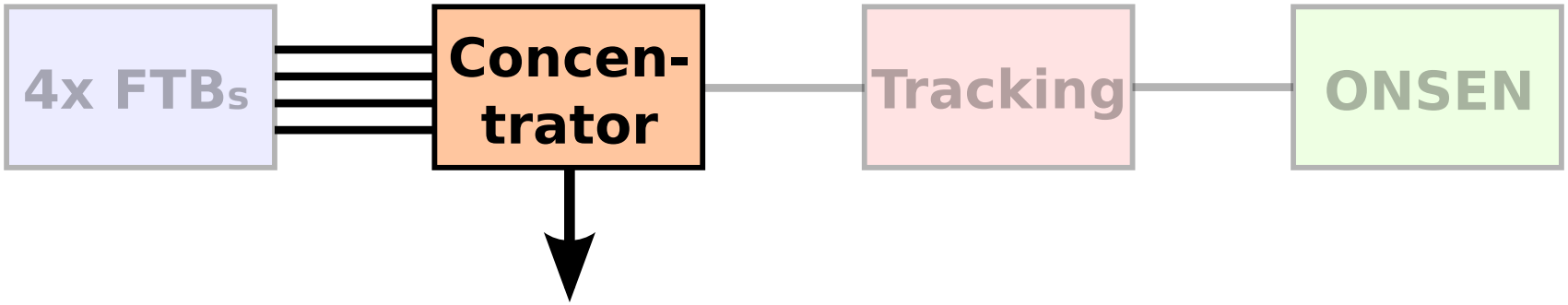
Content

Flow of Data I: FTB



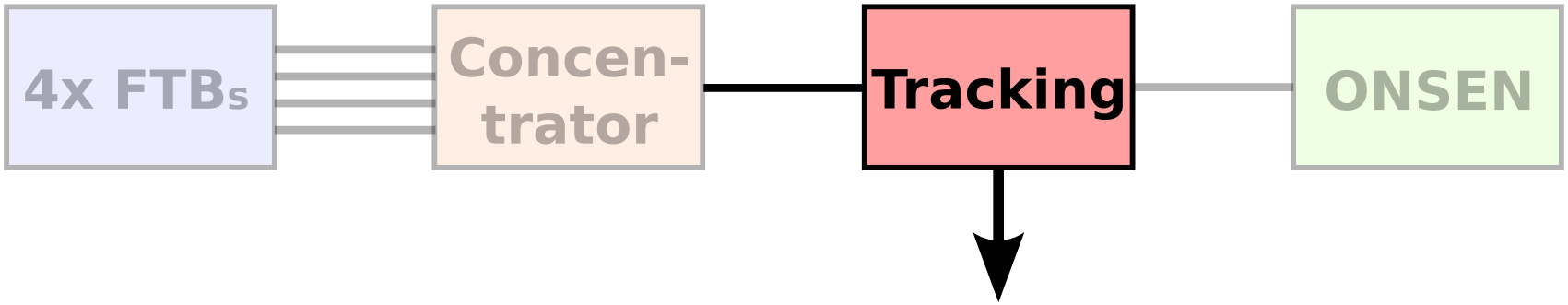
- Finesse Transmitter Board (FTB) sends sampled strip data from Flash ADC (FADC)
- Four different modes: Only two interesting for us
- Zero-Suppressed: 6 consecutive samples for each strip
  - Find peak and peak time with deconvolution function
  - Currently only for the testbeam (maximum sample search)
- Zero-Suppressed+Hit-time: One peak sample and peak time (not for testbeam)

Flow of Data II: DATCON Concentrator



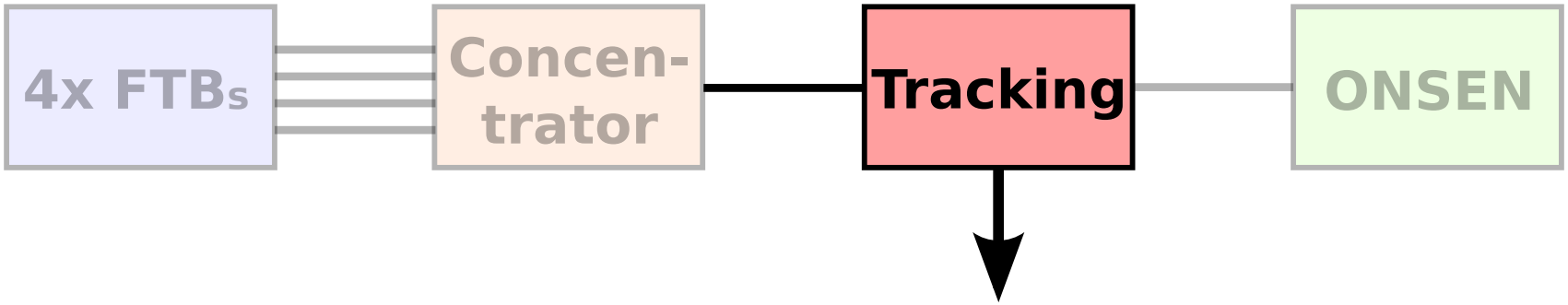
- Preprocessing of SVD strip data and decode of the FTB-FADC protocol
- Cluster engine:
  - Using next neighbour method and a simple center of gravity lookup table.
  - Clusters larger than 4 strips will be split up.
  - Better would be Eta-function method
- Coordinate of cluster

Flow of Data III: Tracking



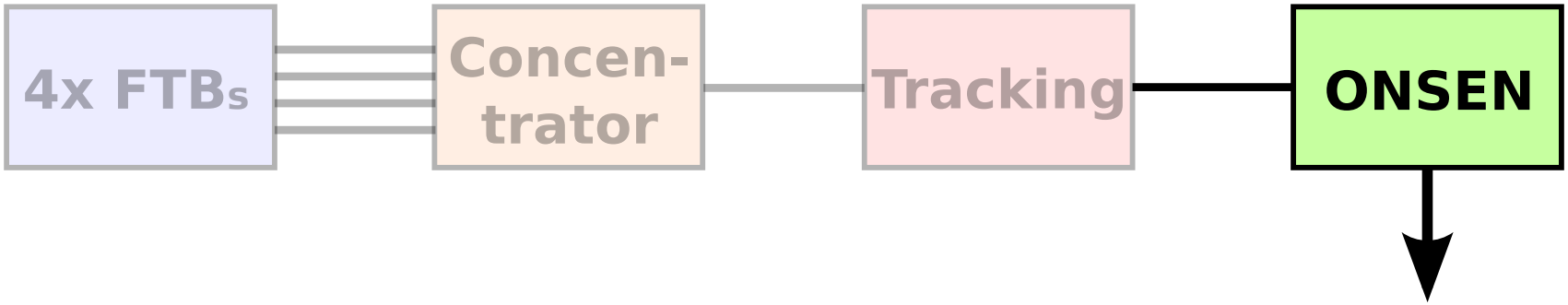
- Receives coordinate data from all Concentrator cards and sort by layer into different memory
- Track reconstruction and fitting in two different steps:
  - r-ϕ: Conformal and Hough transformation
  - y-z: Hough transformation
- Results of Hough transformation provides direct track (helix) parameters

Flow of Data III: Tracking



- With provided track parameters extroplate most probable hit on the two different PXD layers
- Use the reverse coordinate translation engine to obtain the pixel ID for every extrapolated hit
- Define the size of the ROI by error calculation (considering cluster size of the source track samples, momentum of the reconstructed track, Hough sector size...)

Flow of Data IV: ONSEN



- ONline SElector Node (ONSEN) receives ROI information and DHH ID from HLT and DATCON
- ROI: Defined as two opposite Pixel IDs
- Check if incoming Pixel IDs are within the ROI and transmit it to the DAQ

Hough Transformation Basics

- Tracking is based on Fast Hough Transformation
- Hough Transformation is able to find and fit straight tracks

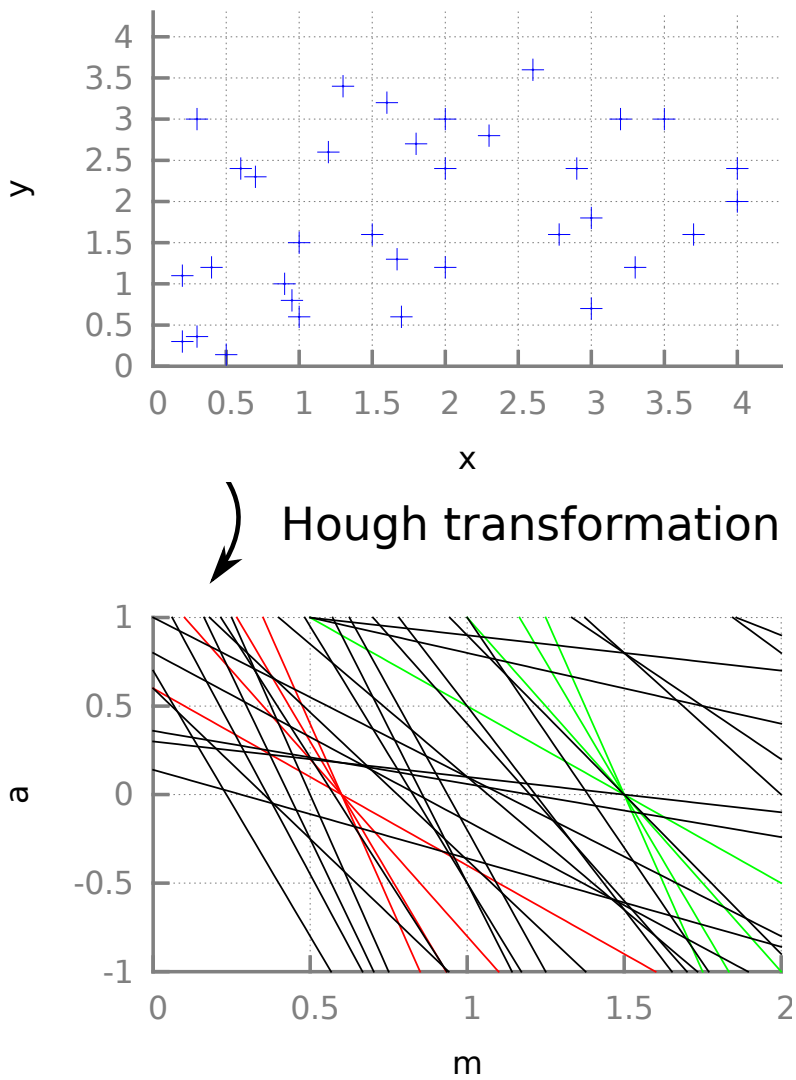
Simple Hough Transformation

$$y_i = m \cdot x_i + a \xrightarrow[trafo]{Hough} a = -m \cdot x_i + y_i$$

- Also works for arc tracks after conformal transformation

Conformal Transformation

$$x' = \frac{x}{(x-x_n)^2+(y-y_n)^2}$$
$$y' = \frac{y}{(x-x_n)^2+(y-y_n)^2}$$



Hough Transformation Basics

- Tracking is based on Fast Hough Transformation
- Hough Transformation is able to find and fit straight tracks

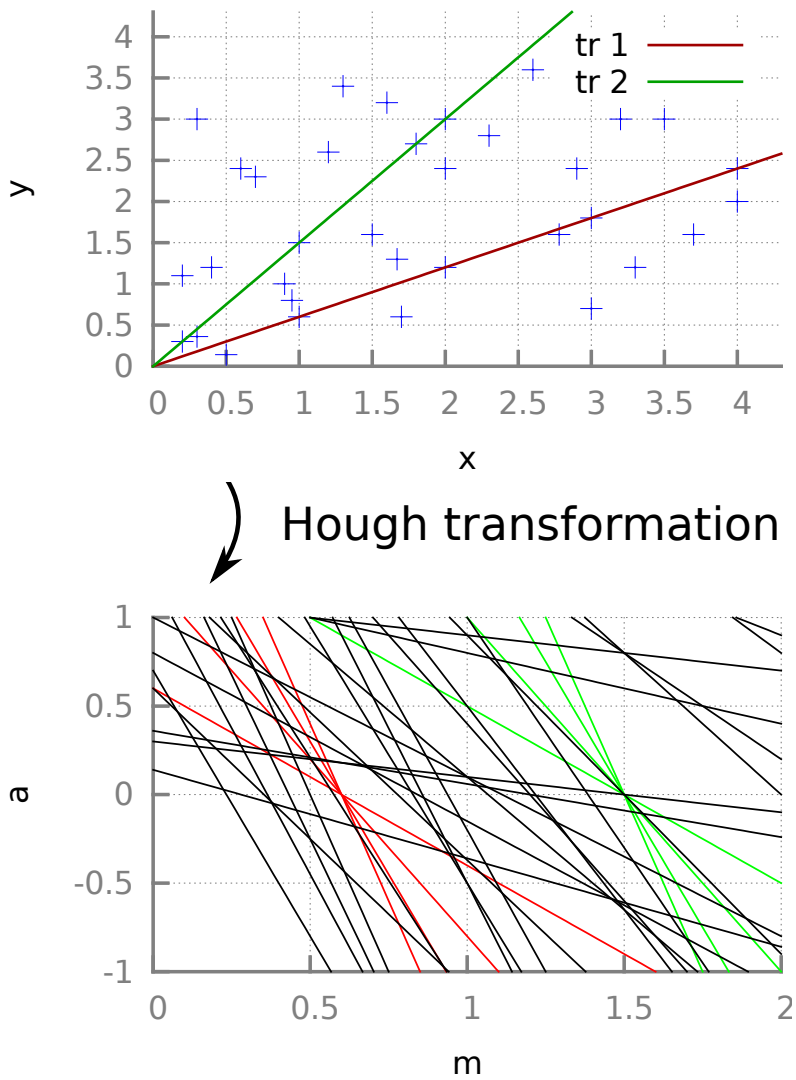
Simple Hough Transformation

$$y_i = m \cdot x_i + a \xrightarrow[trafo]{Hough} a = -m \cdot x_i + y_i$$

- Also works for arc tracks after conformal transformation

Conformal Transformation

$$x' = \frac{x}{(x-x_n)^2+(y-y_n)^2}$$
$$y' = \frac{y}{(x-x_n)^2+(y-y_n)^2}$$

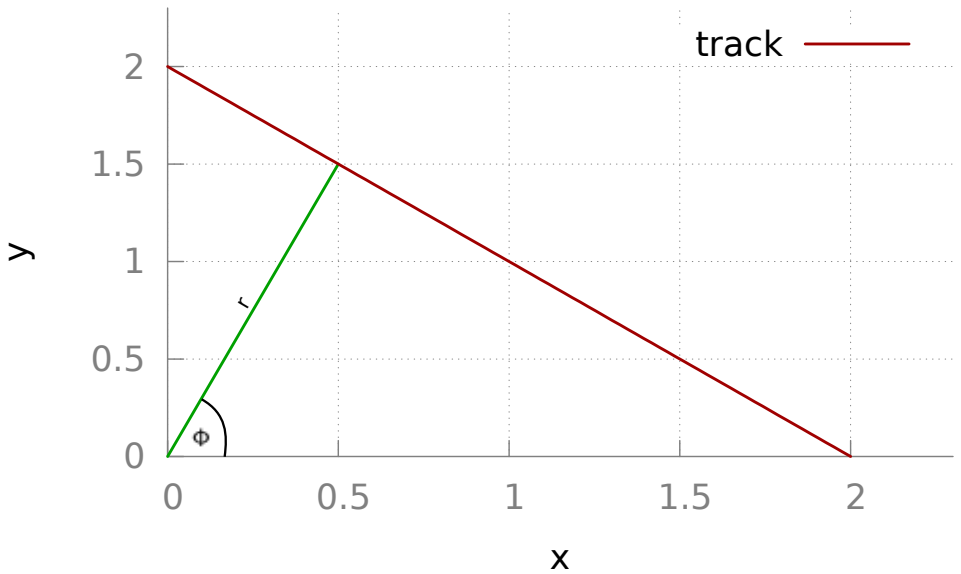


New Parametrization

- Problem at $m \to \infty$; Use other parametrization based on trigonometric function (Hess space)

Hess Transformation

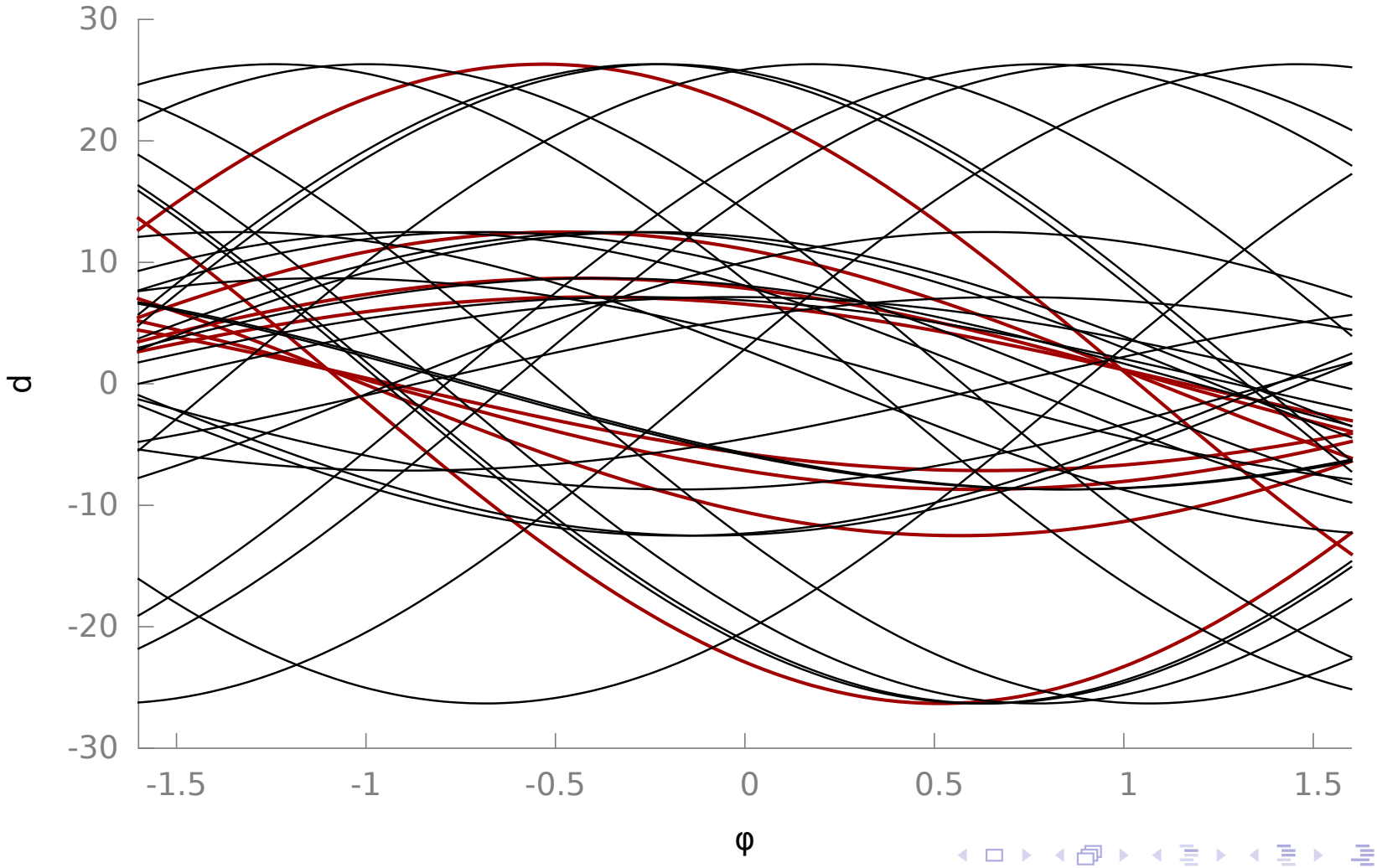
$$d = x_i \cdot cos(\Phi) + y_i \cdot sin(\Phi)$$



- Divide and conquer type algorithm

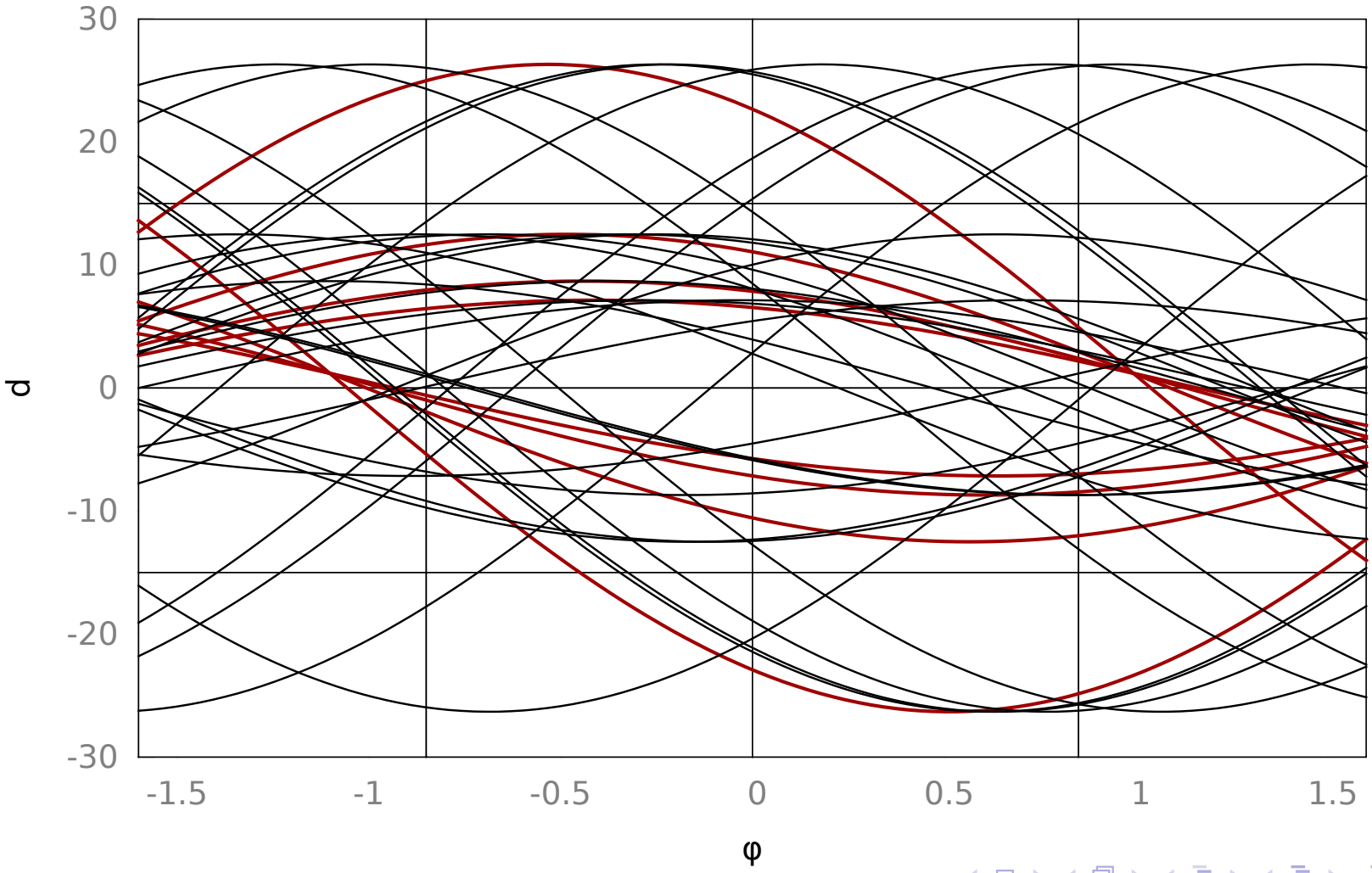
Realistic Example

- Hough space with new parametrization



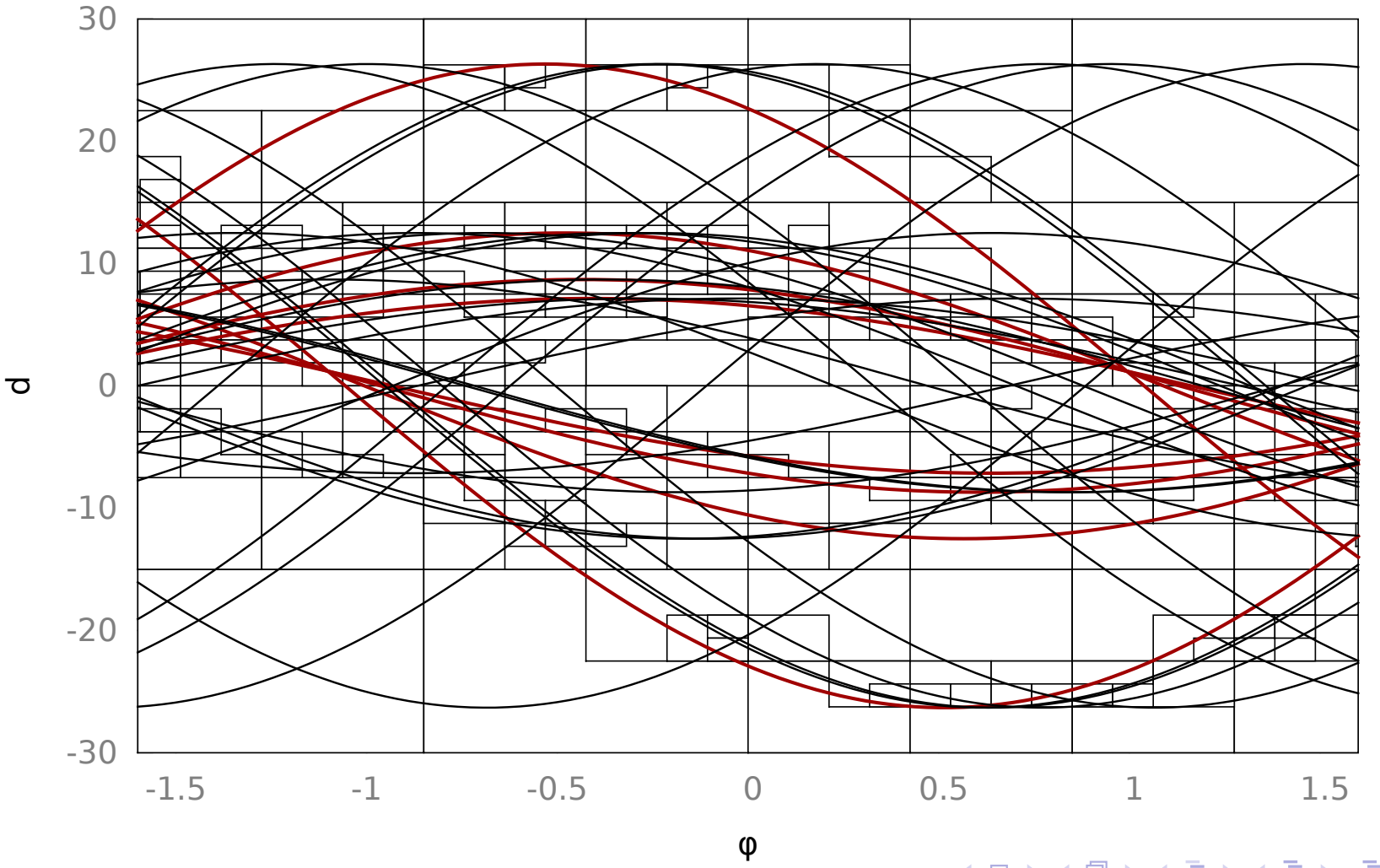
Realistic Example

- Divide into sectors and count number of tracks



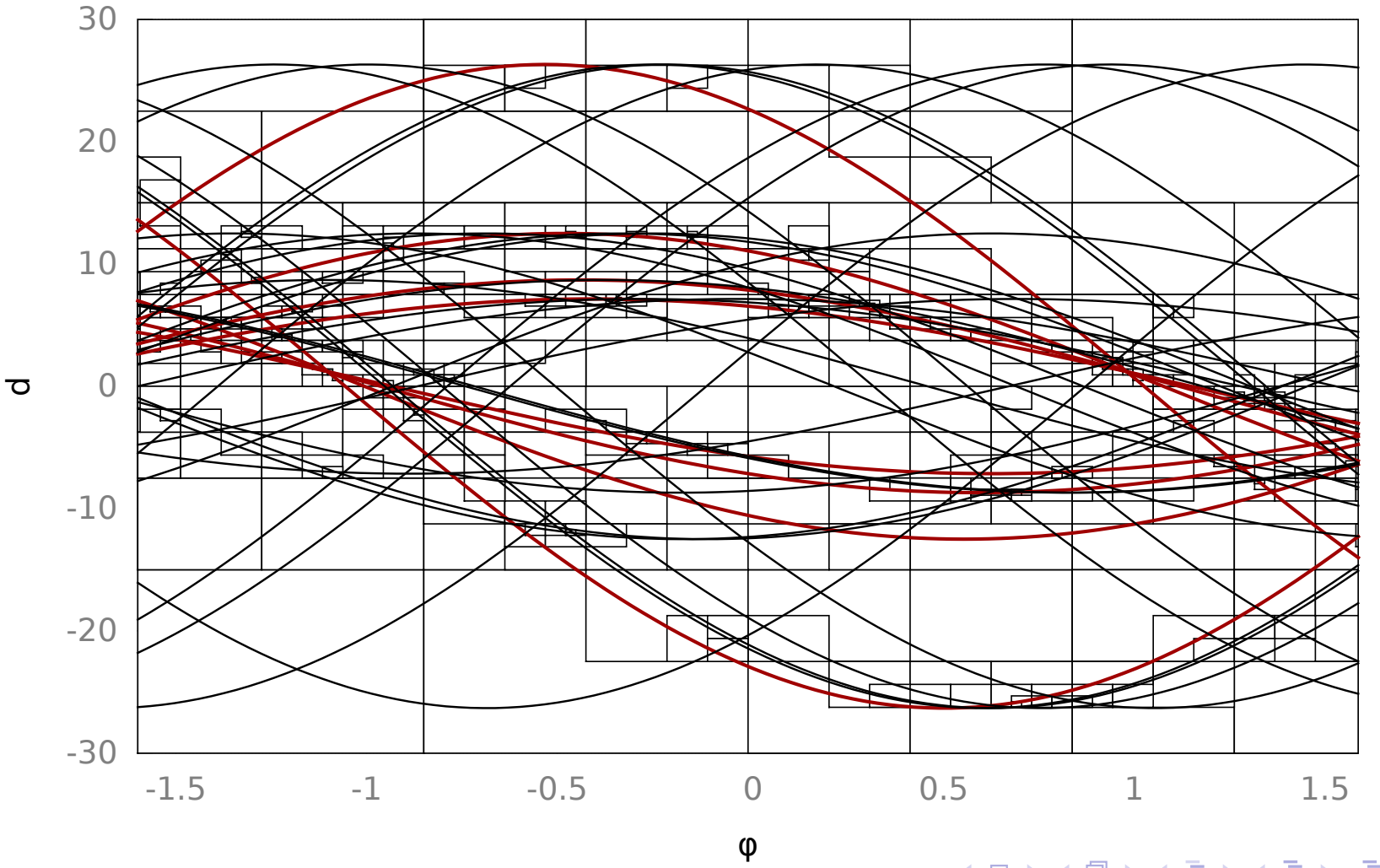
Realistic Example

- Iteration 4



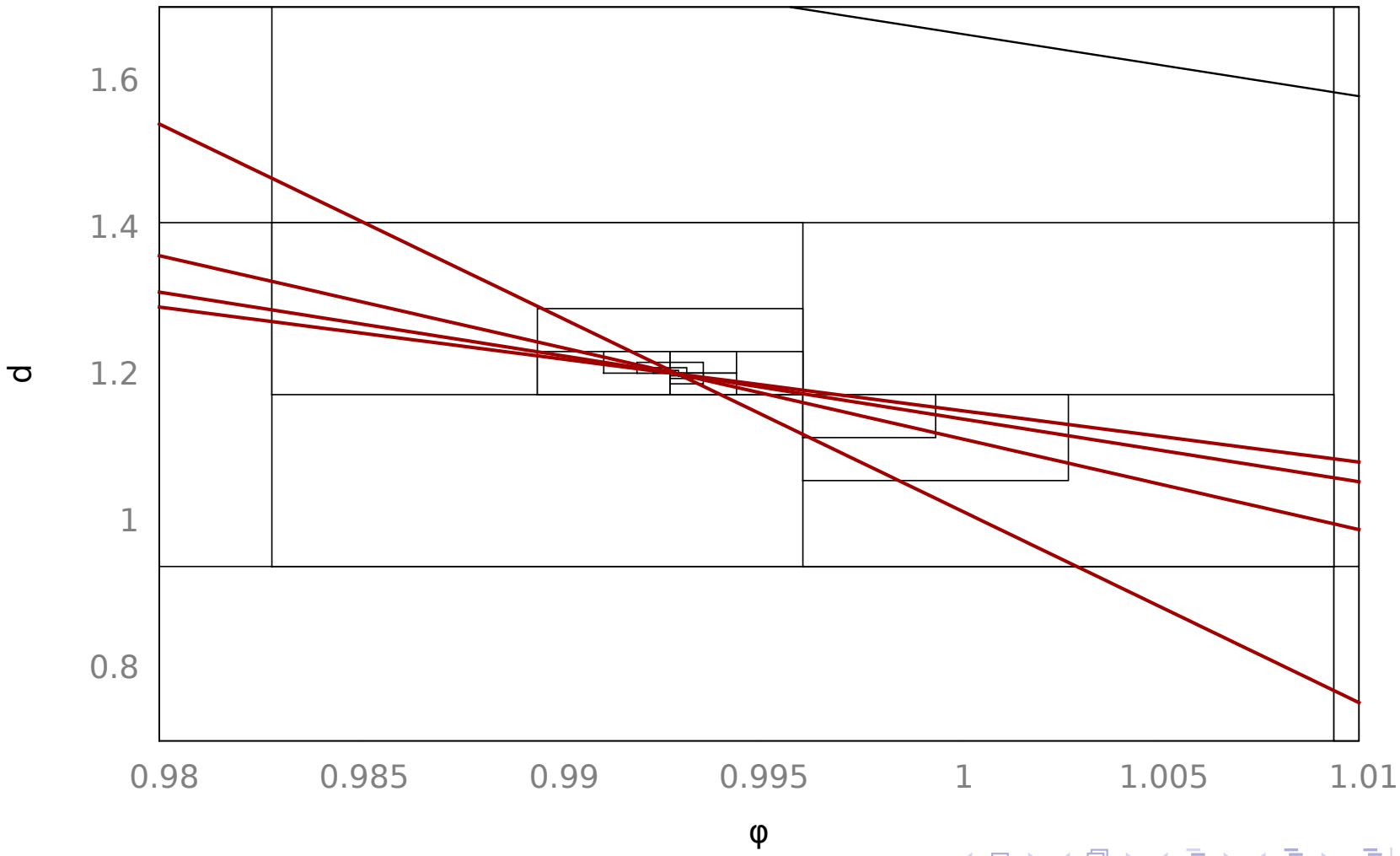
Realistic Example

- Iteration 8



Realistic Example

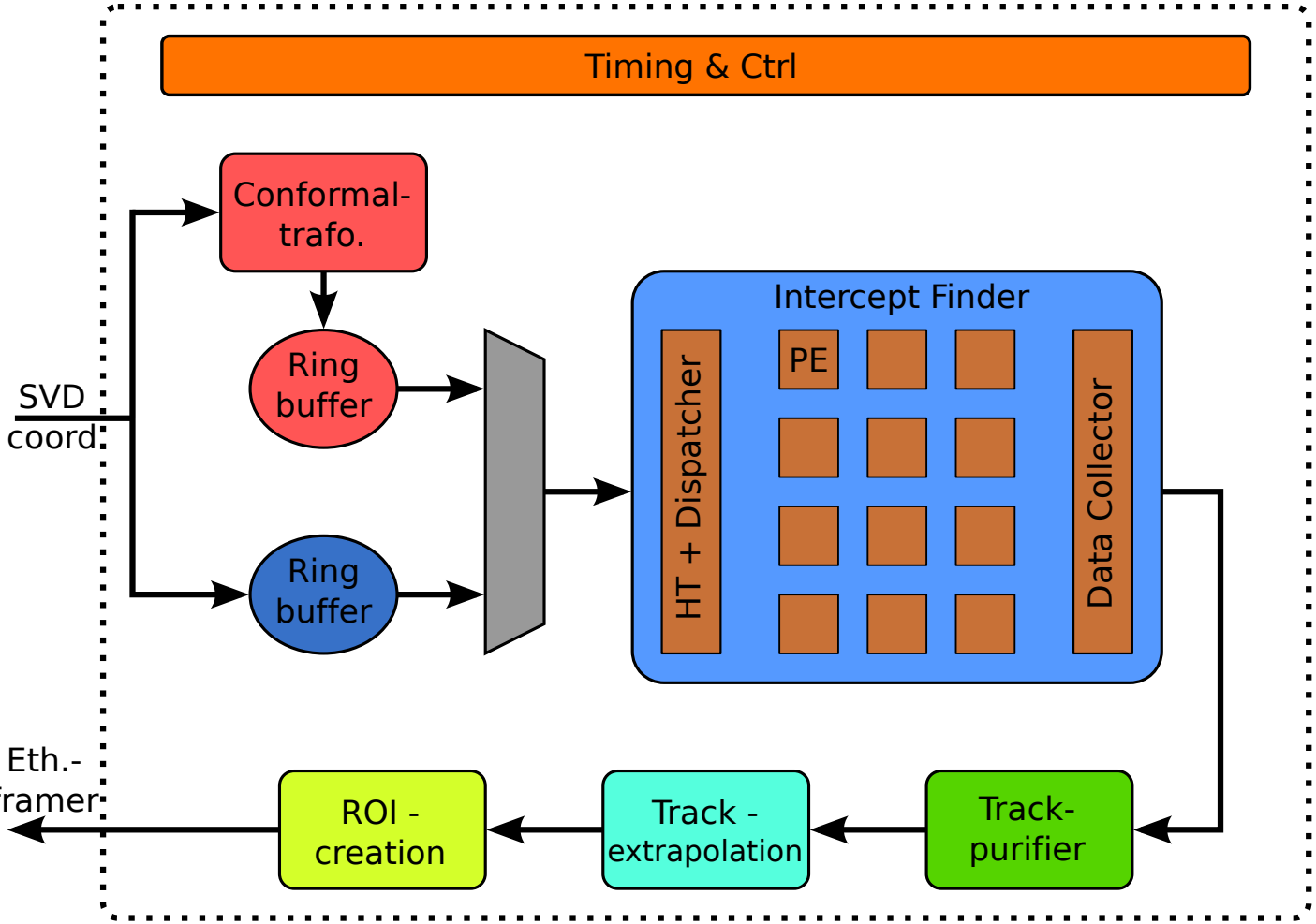
- Zoom



Pros and Cons

- Track finding and fitting in one step
- Size of rectengular as measure for error (later important for calculating ROI size)
- Limit Hough space to area with potential for reasonable tracks
- Intersection of lines not always clear due to
  - Unknown exact vertex
  - MS and other physics effects
  - Numerical instability (especially in FPGA)
- Requires high amount of computing resources

FPGA Track Reconstruction Prototype Architecture

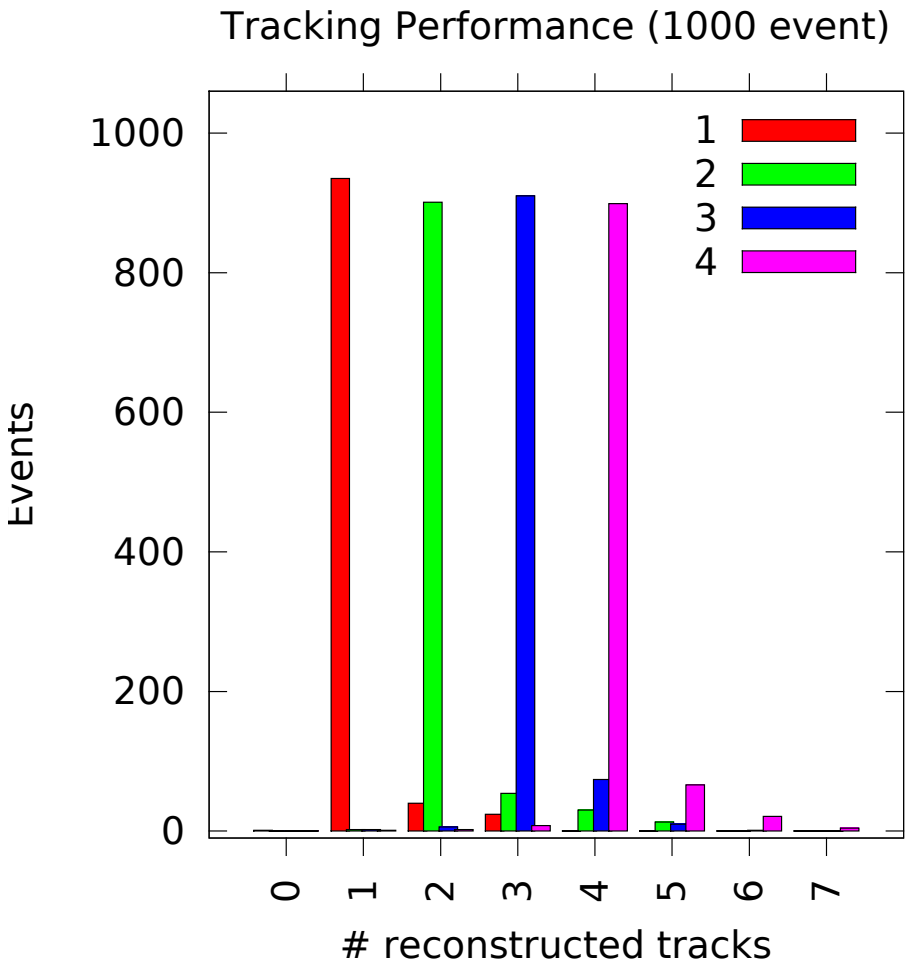


- PE: Processing Element, basic comparator to detect lines in cell
- Track Purifier: Remove duplicated found tracks and combine two 2D-sets

Content

Efficiency Study

Generated 1000 events per tracking study for 1, 2, 3 and 4 tracks:

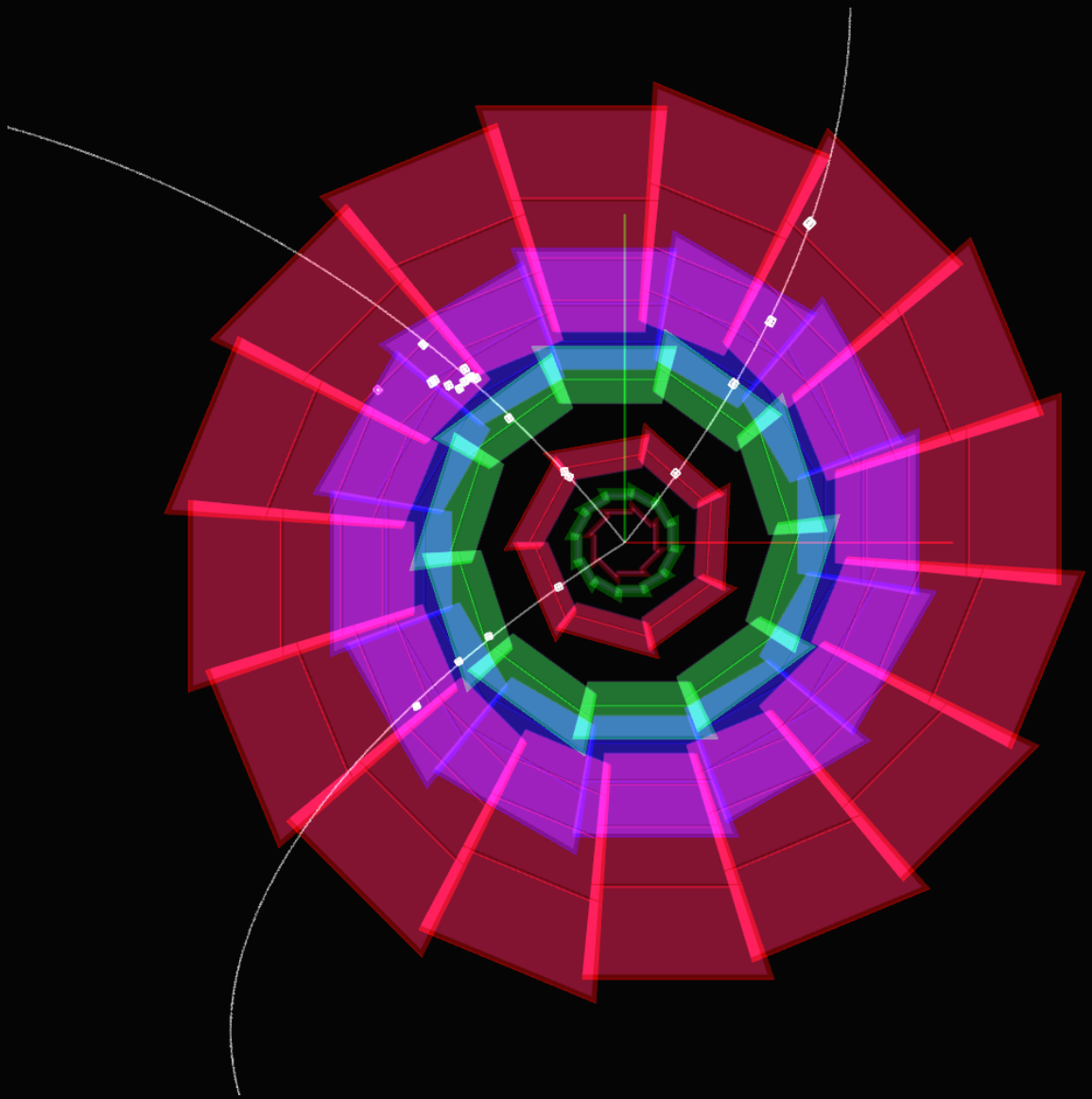


Setup:

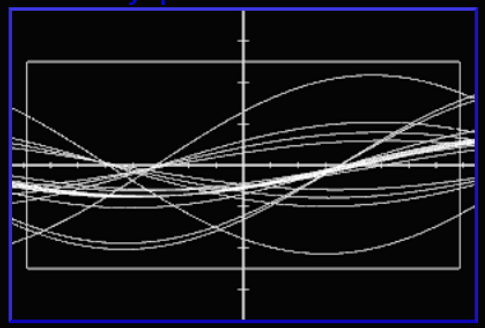
- Full detector simulation (BASF2)
- Generated e^+, e^- momentum: 0.3 to 3 *GeV*
- No background (so far)

Gen. tracks	Reco rate	Fake rate
1	99.8%	8.9%
2	99.6%	15.3%
3	98.2%	9.7%
4	98.5%	10.8%

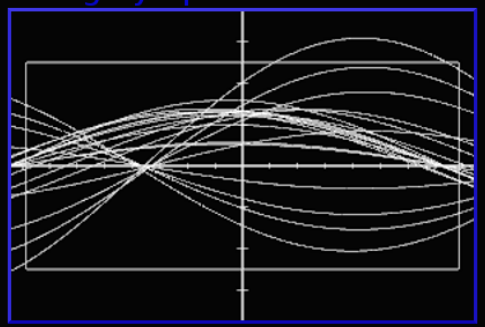
Simulation Example Visualization: 3 Tracks XY



Hess xy-plane



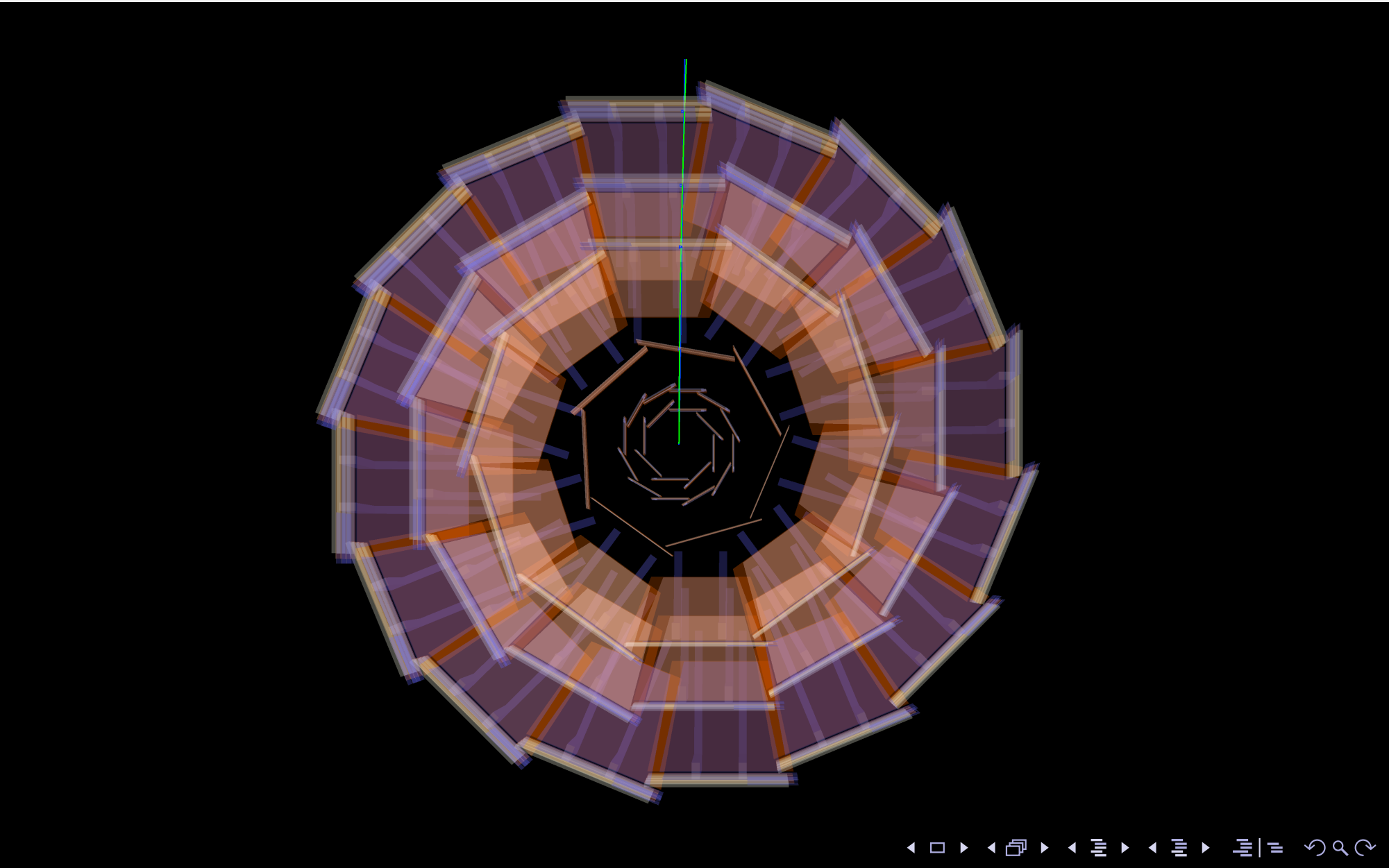
Hough yz-plane



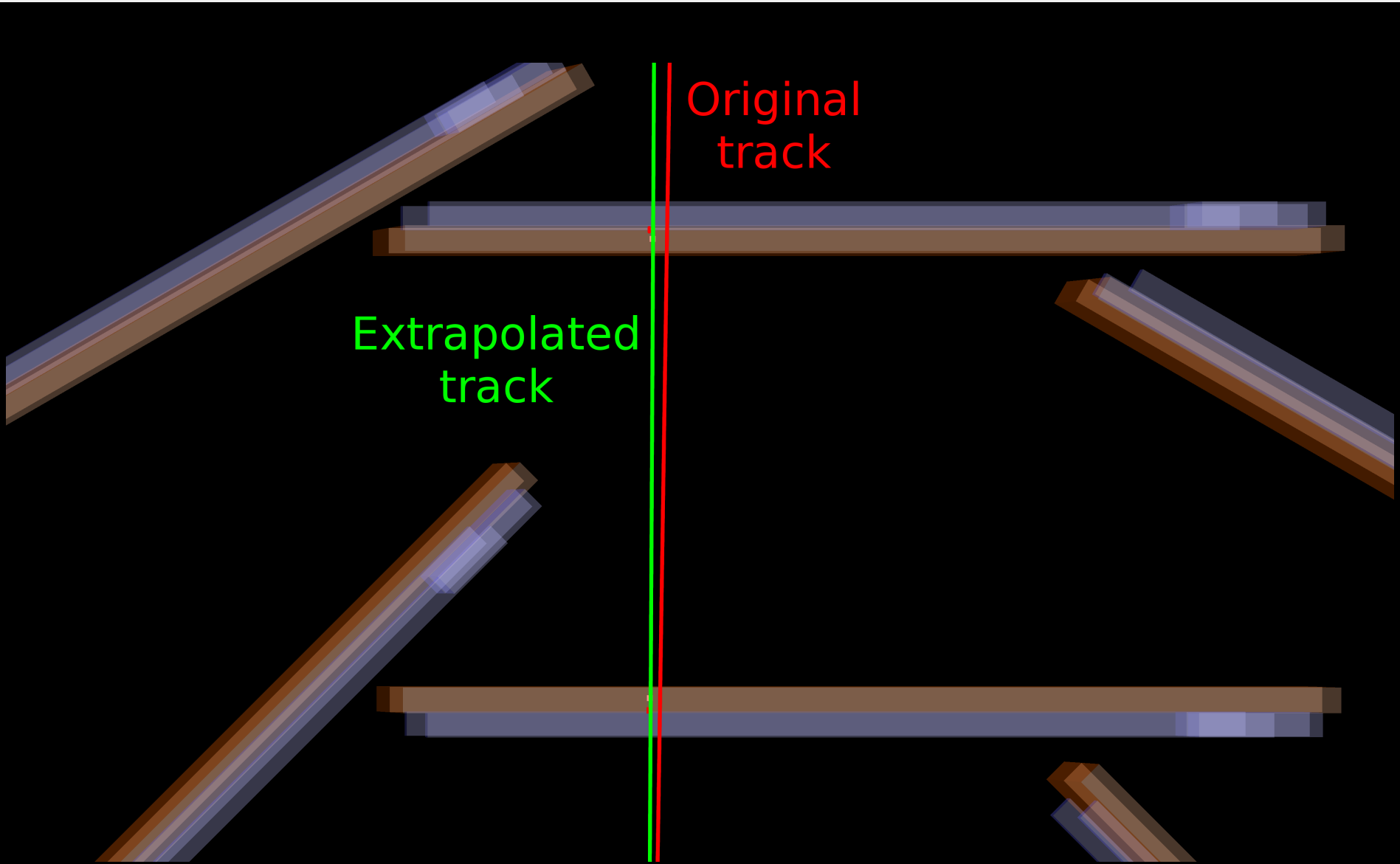
ROI Simulation Result

- basf2 simulation with full detector geometry, no background, uniform distribution of phi and theta (limited between 50 and 120 deg)
- Generating e^+, e^- with momentum from 0.1 to 3 GeV
- Tracking: Hough transformation with 8 iterations in (x,y) and (y,z)
- Extrapolation to PXD ladder with angle/track radius lookup table
- Precise hit-finding by track extrapolation over sampling and point of closest approach

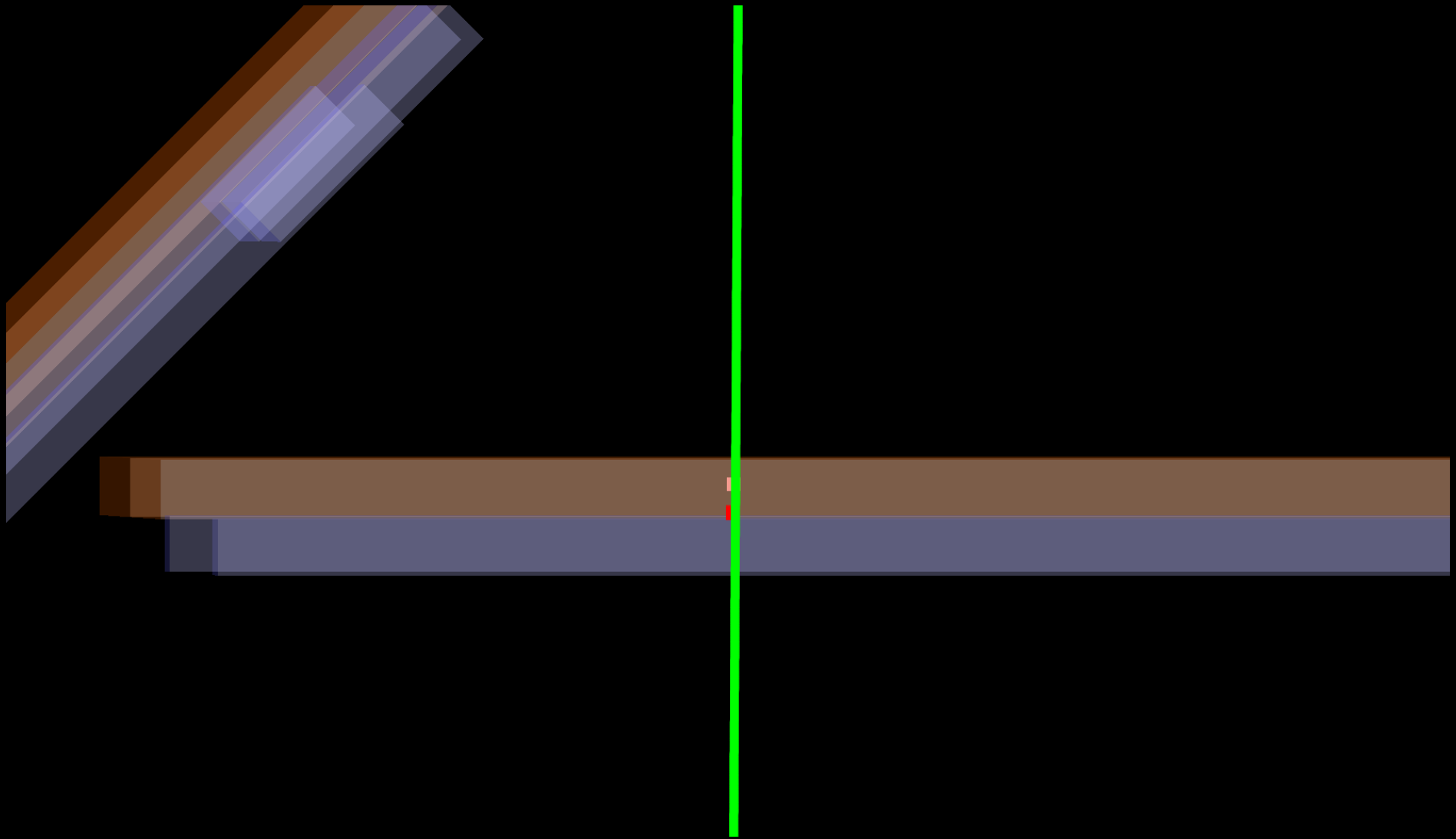
Visualization



Visualization



Visualization



Performance of ROI size

- Table shows difference between real and extrapolated hit
- 10 Events for each momentum, taken worst value
- Layer 1 pixel pitch = 50x60 um ; Layer 2 pixel pitch = 50x85 um

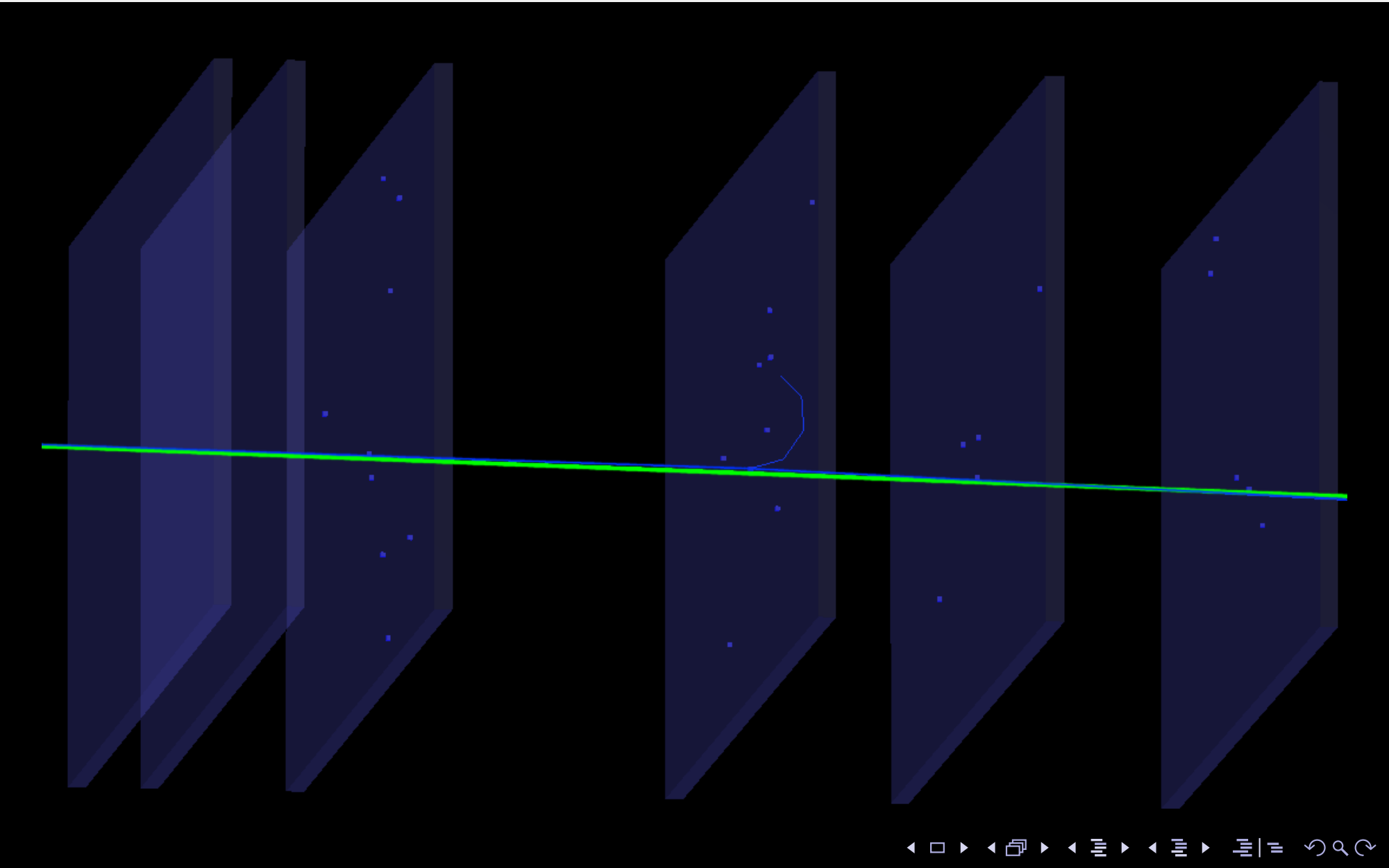
Layer 1		
p [GeV]	Δx [px]	Δz [px]
3	0.65	0.88
1	0.51	0.43
0.6	5.41	0.89
0.3	7.72	0.47
0.1	29.28	17.29

Layer 2		
p [GeV]	Δx [px]	Δz [px]
3	1.11	0.96
1	0.76	0.47
0.6	7.51	0.98
0.3	10.95	0.42
0.1	42.49	17.84

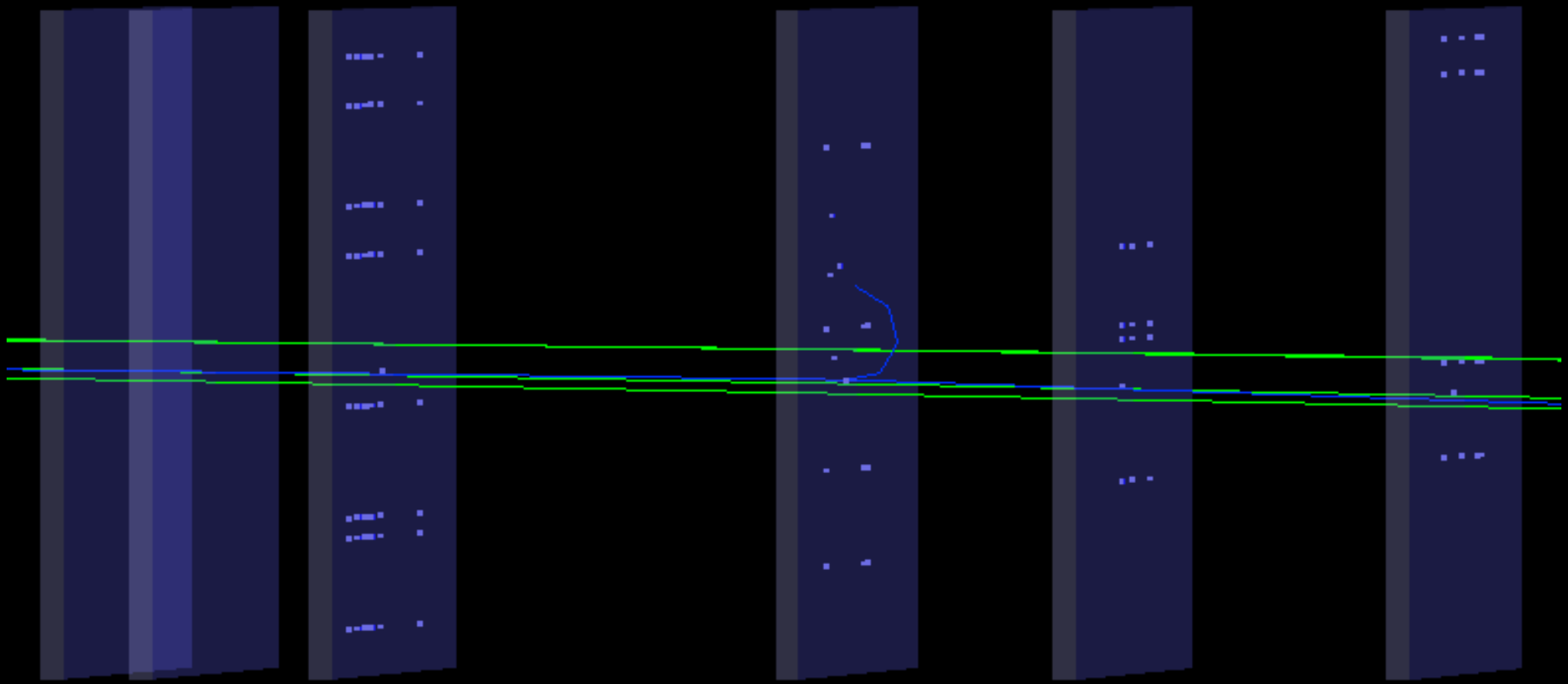
Testbeam Simulation Setup

- basf2 simulation with testbeam geometry
- Generating e^- with momentum from 2 to 6 *GeV*
- Injecting ghost hits from dead strip map*New*
- Tracking: Hough transformation with 7 iterations in (x,y) and (y,z)

Visualization (Good)



Visualization (Bad)



Content

Integration Status

- BASF2 Module in tracking/modules" done
- Includes clustering engine and data format conversion
- Two new dataobjects: "SVDHoughTrack" and "SVDHoughCluster" done
- Adaption to "display" to visualize "SVDHoughTrack" and "SVDHoughCluster" and minor changes to "GeoVXD" for more detailed sensor information output done
- Adaption necessary to the Hough transformation module. Currently stand-alone library written in C. Needs integration into BASF2 framework or in externals? todo

Conclusion and Future Plans

- Hough Transformation do finding and fitting at the same time
- With increasing background number of fake tracks greatly increases → Filters and pre-processing useful
- FPGA limited to fixed point numbers
- ROI extrapolation works as expected

Plans:

- Integration into basf2 ongoing (but not before testbeam in Januar)
- Investigation of other suited algorithm required (for pre-filtering)
- Redo simulation with larger statistics and background

Thank you for your attention!

Porting: General statements

- In scripts don't use "/bin/sh" or "/bin/bash", but "/usr/bin/env bash"
- Write ...bash, when the script uses bash specific features
- No linux specific low-level interfaces like pfctl
- Some minor changes to geant4 (missing header) and gtest (missing define) are required
- Use variables in Makefiles for programs like sed, make etc...