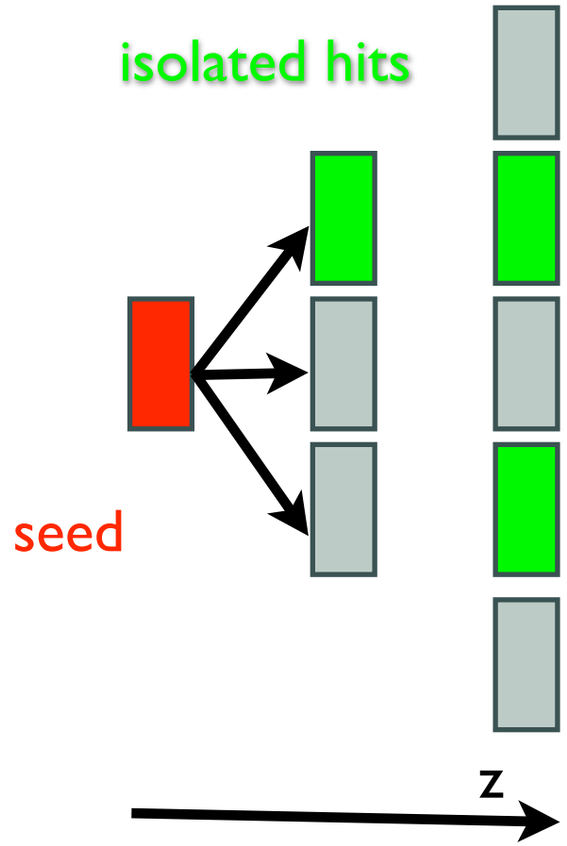
Track Segment Analysis update

L. Weuste
Max Planck Institut für Physik
Excellence Cluster „Universe“
München

CALICE Collaboration Meeting
Shinshu University - March 2012

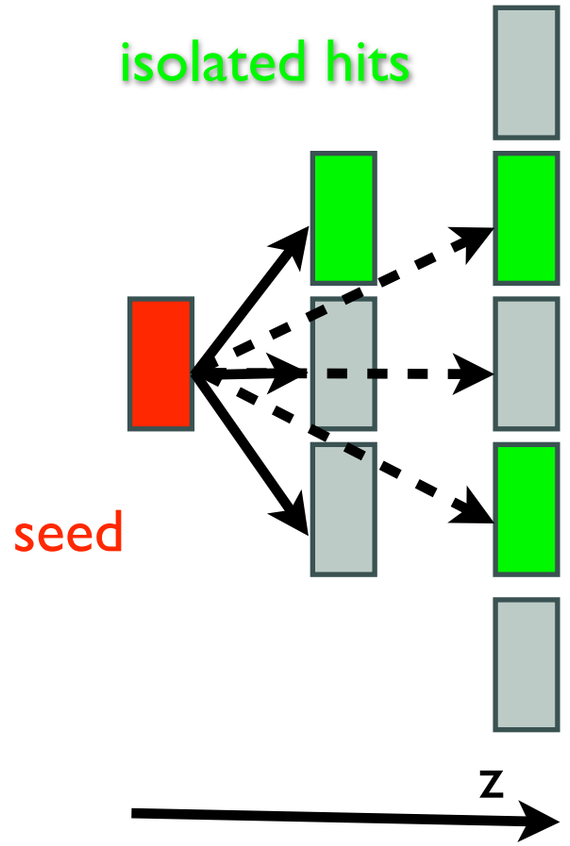
Tracking in the AHCal

- Already presented in CAN-022
  - Nearest Neighbour algorithm
  - Needs 1 hit per layer
  - Based on layer isolated hits,
    i.e. hits with no adjacent hits in the same layer
- Plan: Publication (JINST?)
- Rewrite of code
  - No fundamental changes
  - Usage of official geometry classes
  - Made algorithm more general
    - Completely recursive implementation
    - With simplification: No need for special treatment of certain geometric cases
    - Improving identification of inclined tracks with gaps



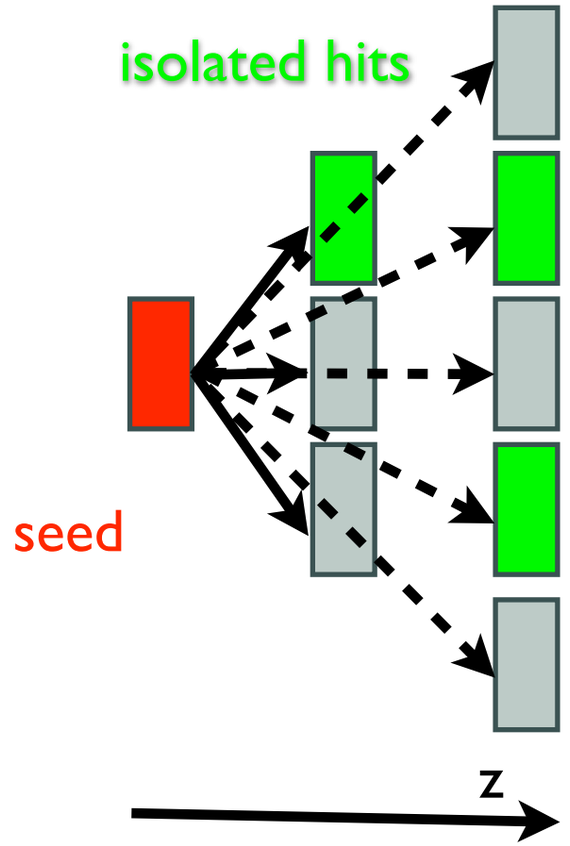
Tracking in the AHCal

- Already presented in CAN-022
  - Nearest Neighbour algorithm
  - Needs 1 hit per layer
  - Based on layer isolated hits,
    i.e. hits with no adjacent hits in the same layer
- Plan: Publication (JINST?)
- Rewrite of code
  - No fundamental changes
  - Usage of official geometry classes
  - Made algorithm more general
    - Completely recursive implementation
    - With simplification: No need for special treatment of certain geometric cases
    - Improving identification of inclined tracks with gaps

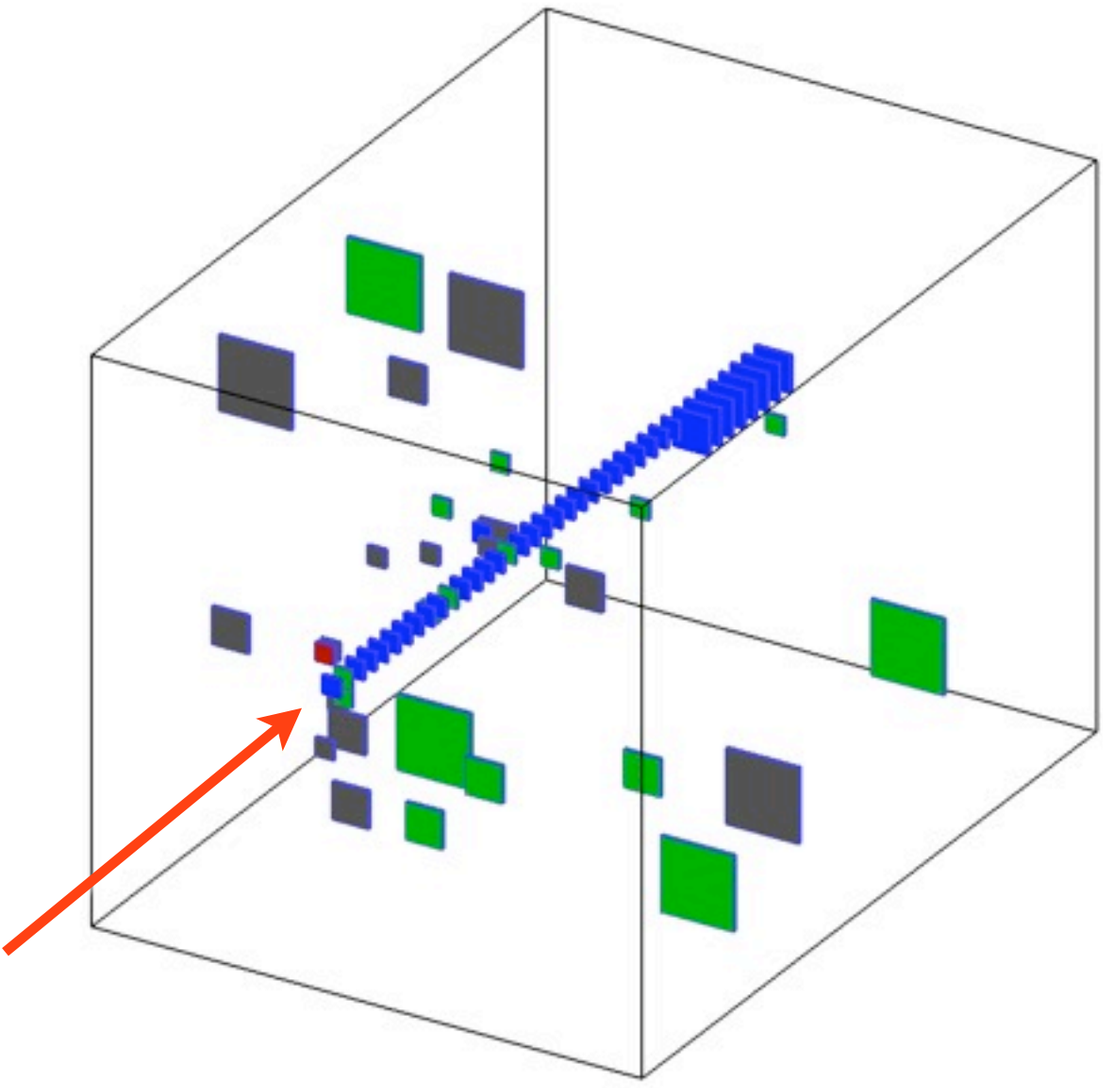


Tracking in the AHCal

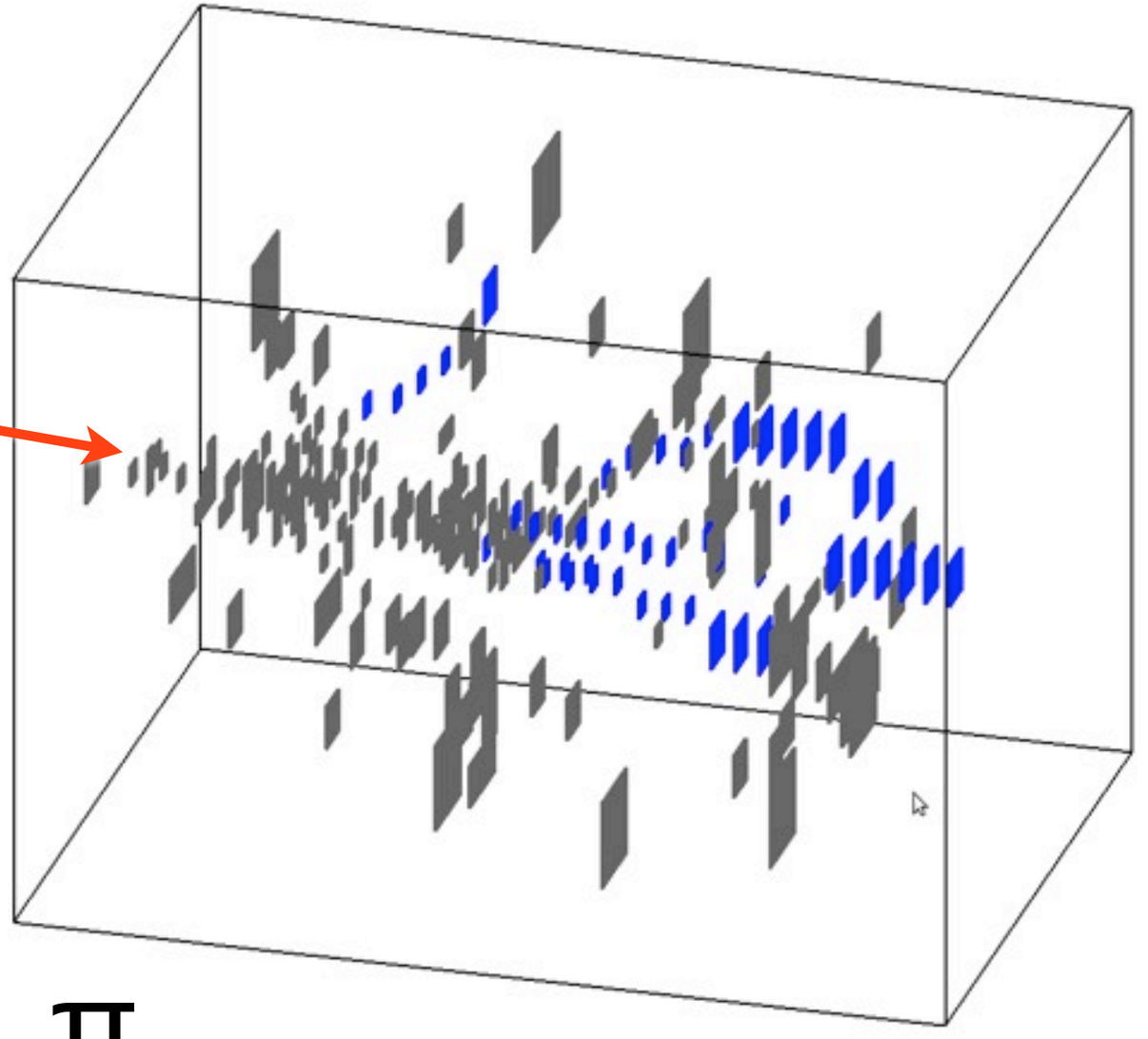
- Already presented in CAN-022
  - Nearest Neighbour algorithm
  - Needs 1 hit per layer
  - Based on layer isolated hits,
    i.e. hits with no adjacent hits in the same layer
- Plan: Publication (JINST?)
- Rewrite of code
  - No fundamental changes
  - Usage of official geometry classes
  - Made algorithm more general
    - Completely recursive implementation
    - With simplification: No need for special treatment of certain geometric cases
    - Improving identification of inclined tracks with gaps



Example events

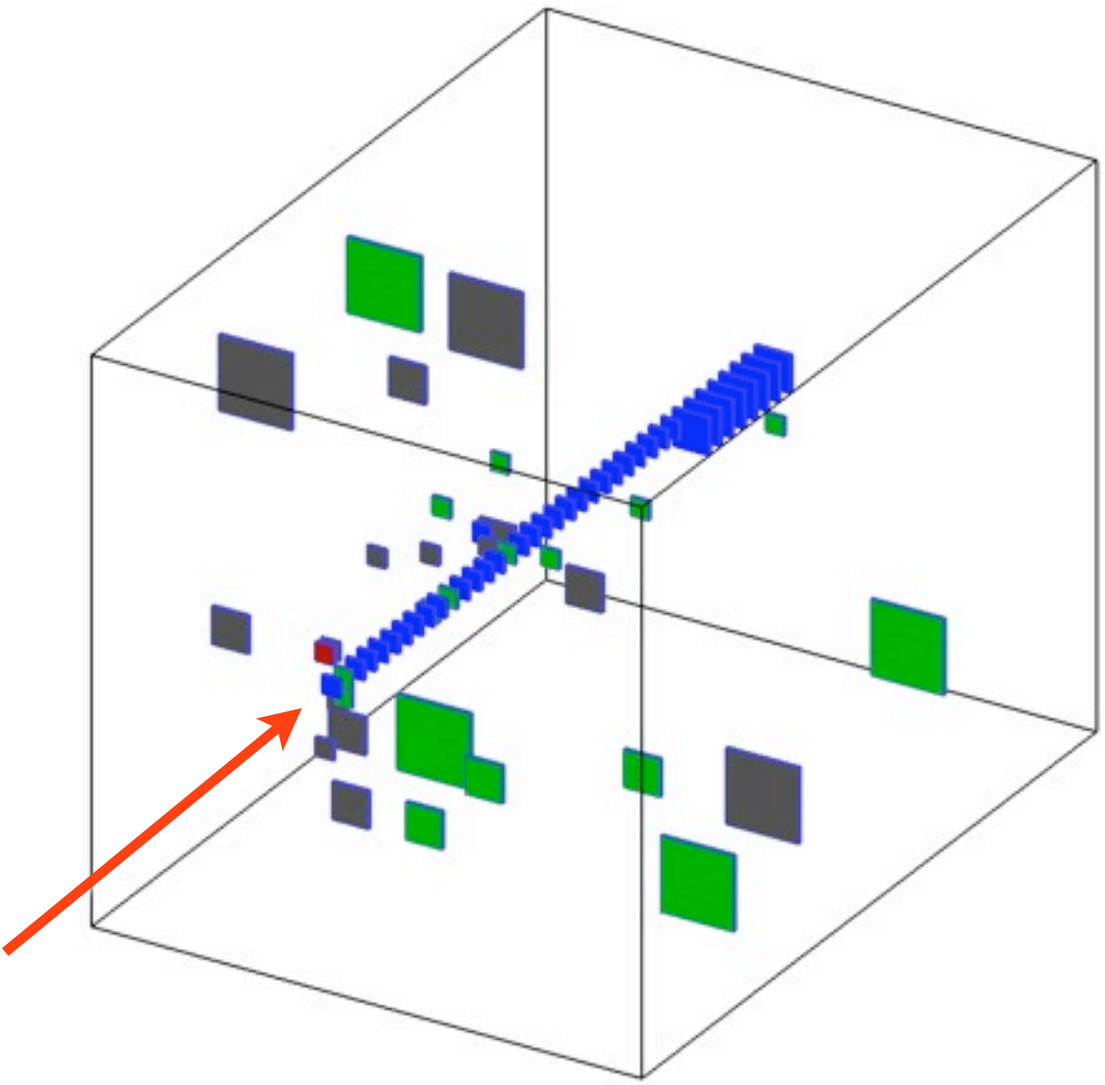


μ

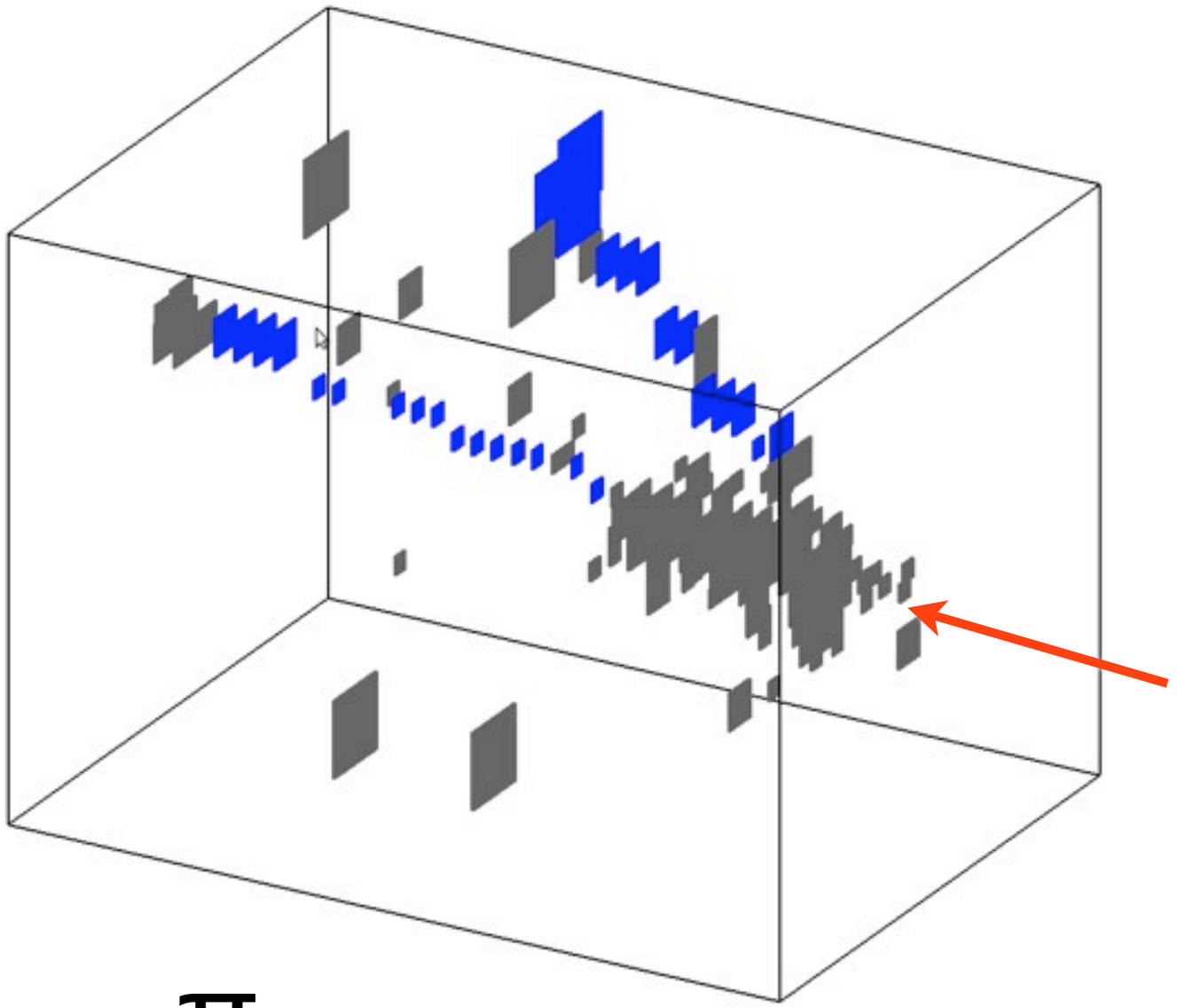


π

Example events



μ



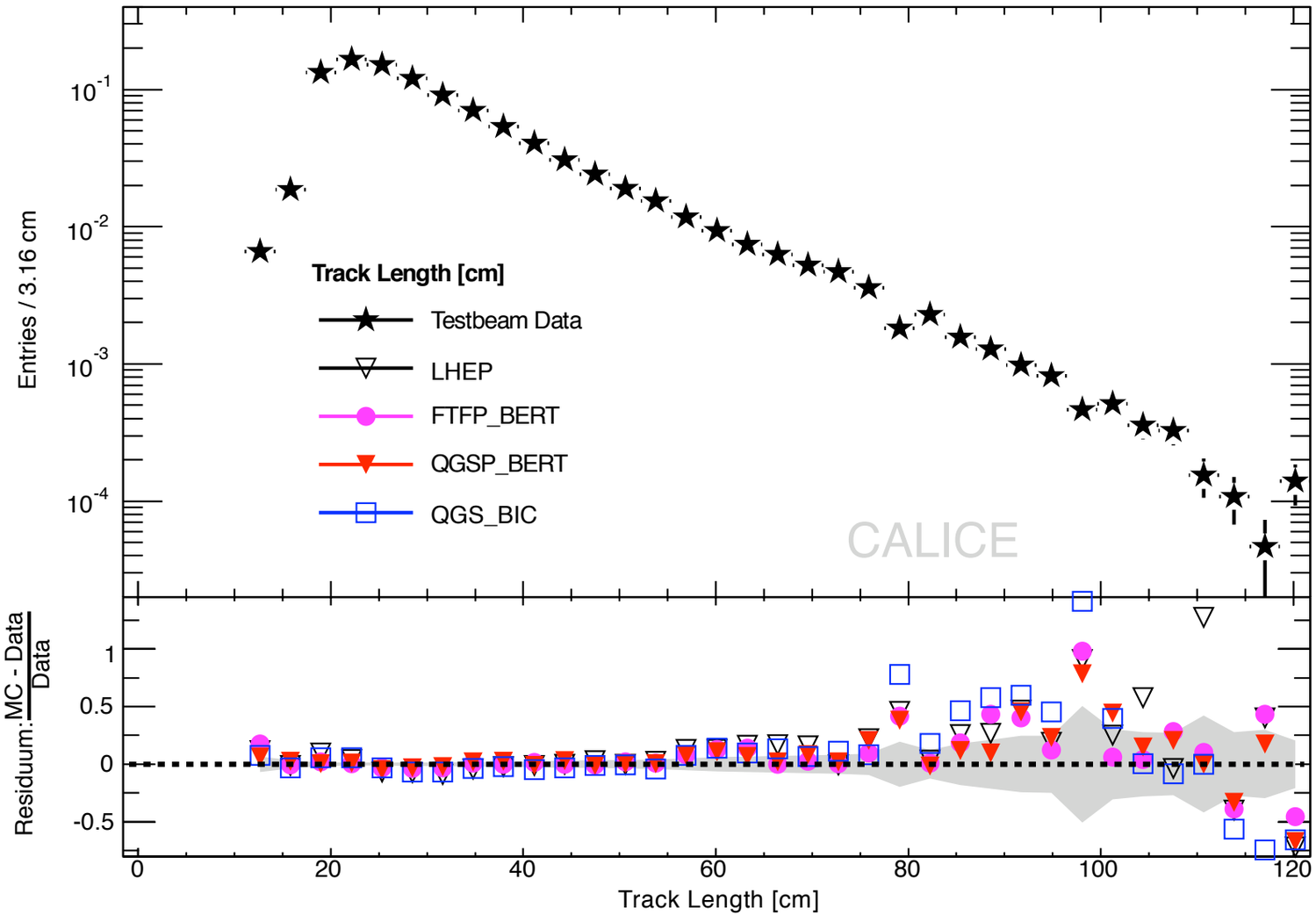
π

Example usage: track length

- Track length `l`
  - λ_{track} ≡ slope of exponential fit („typical track length“)
- Efficiency of track finder ≠100% ==> $\lambda_{track} \neq \lambda$
  - Toy MC to show effect

$$l(x) = l_0 \cdot \exp(-\frac{x}{\lambda_{\mathrm{track}}})$$

Efficiency = abort prob / layer

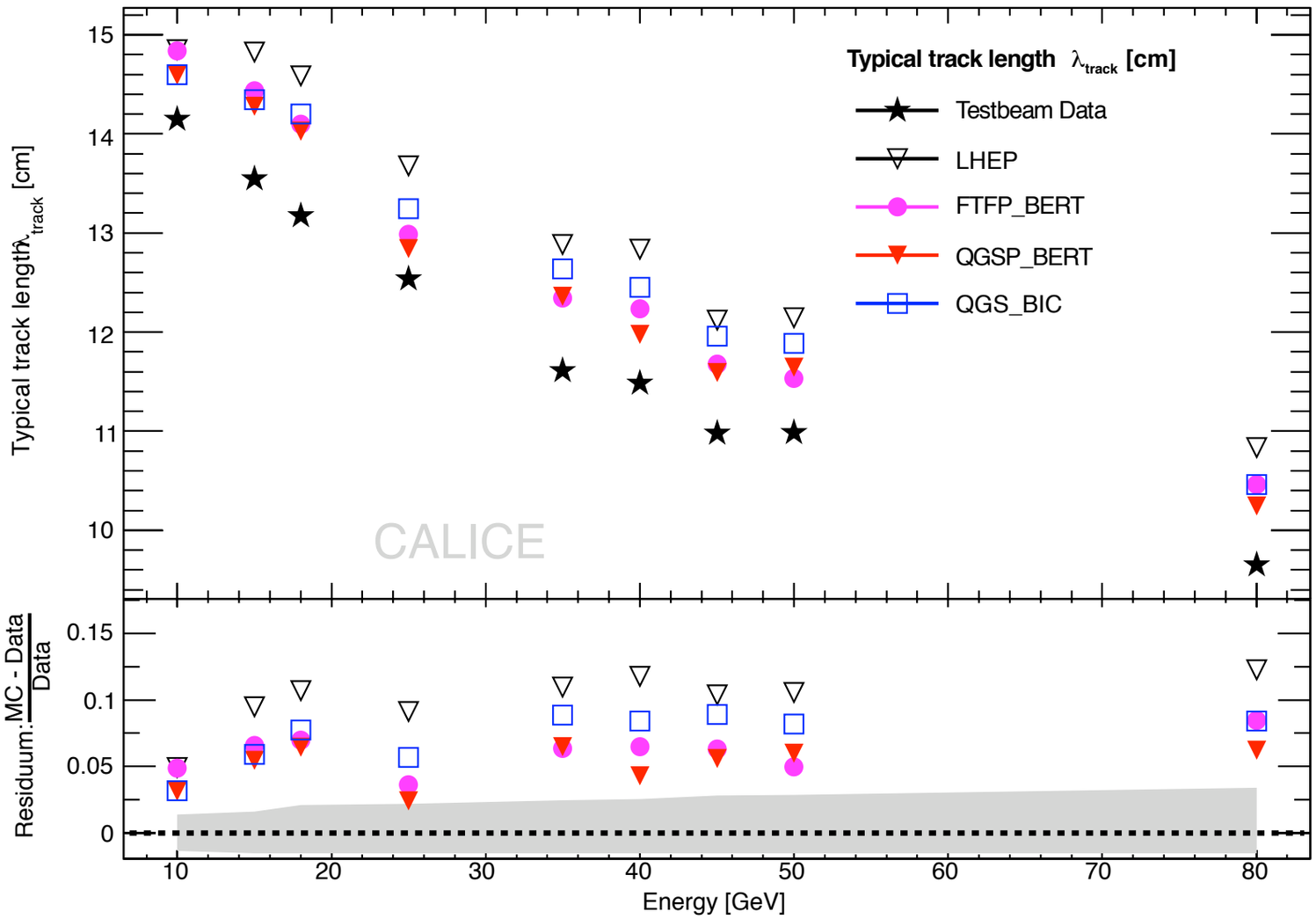


Example usage: track length

- Track length `l`
  - λ_{track} ≡ slope of exponential fit („typical track length“)
- Efficiency of track finder ≠100% ==> $\lambda_{track} \neq \lambda$
  - Toy MC to show effect

$$l(x) = l_0 \cdot \exp(-\frac{x}{\lambda_{\mathrm{track}}})$$

Efficiency = abort prob / layer

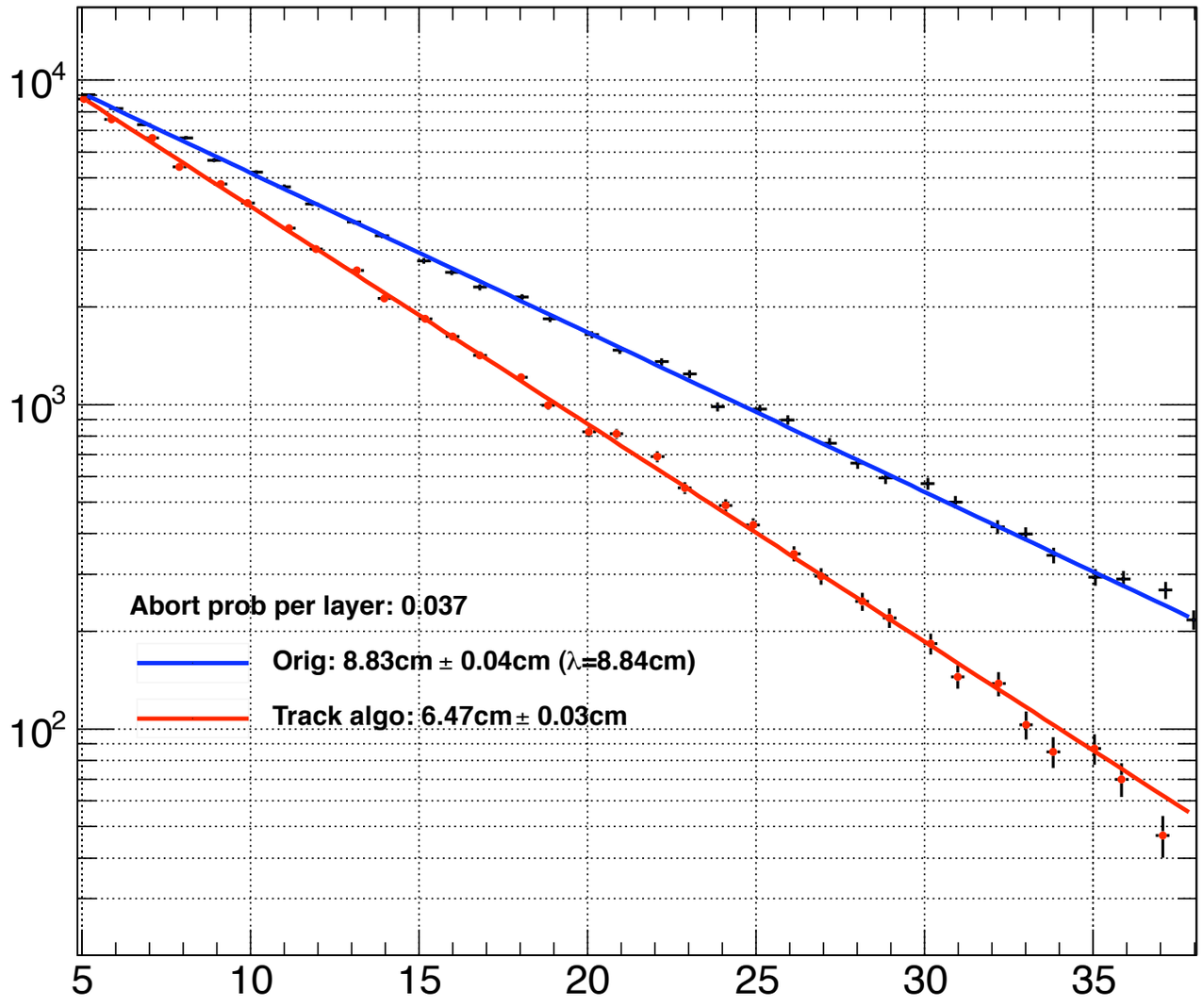


Example usage: track length

- Track length `l`
  - λ_{track} ≡ slope of exponential fit („typical track length“)
- Efficiency of track finder ≠100% ==> $\lambda_{track} \neq \lambda$
  - Toy MC to show effect

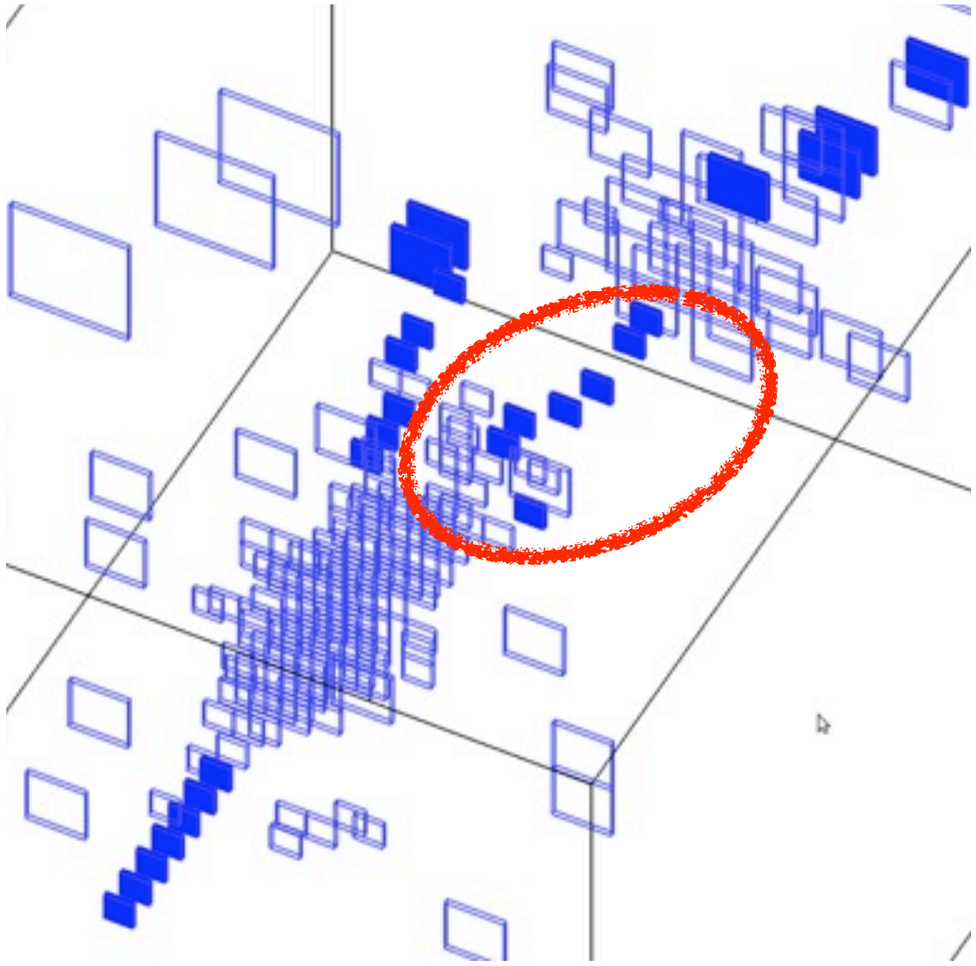
$$l(x) = l_0 \cdot \exp(-\frac{x}{\lambda_{\mathrm{track}}})$$

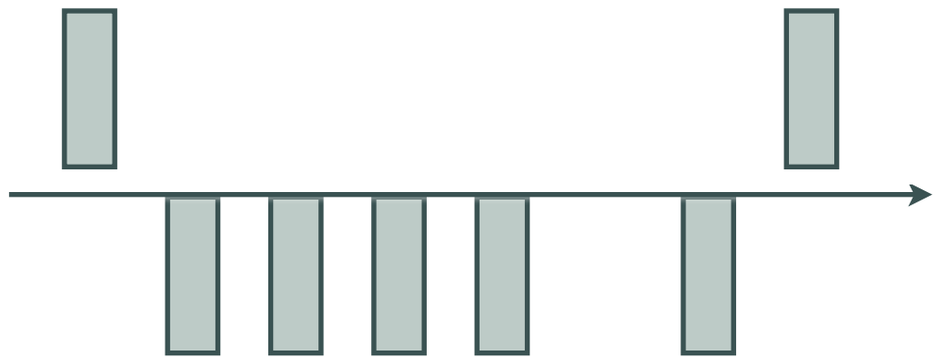
Efficiency = abort prob / layer



Track finding: Imperfections

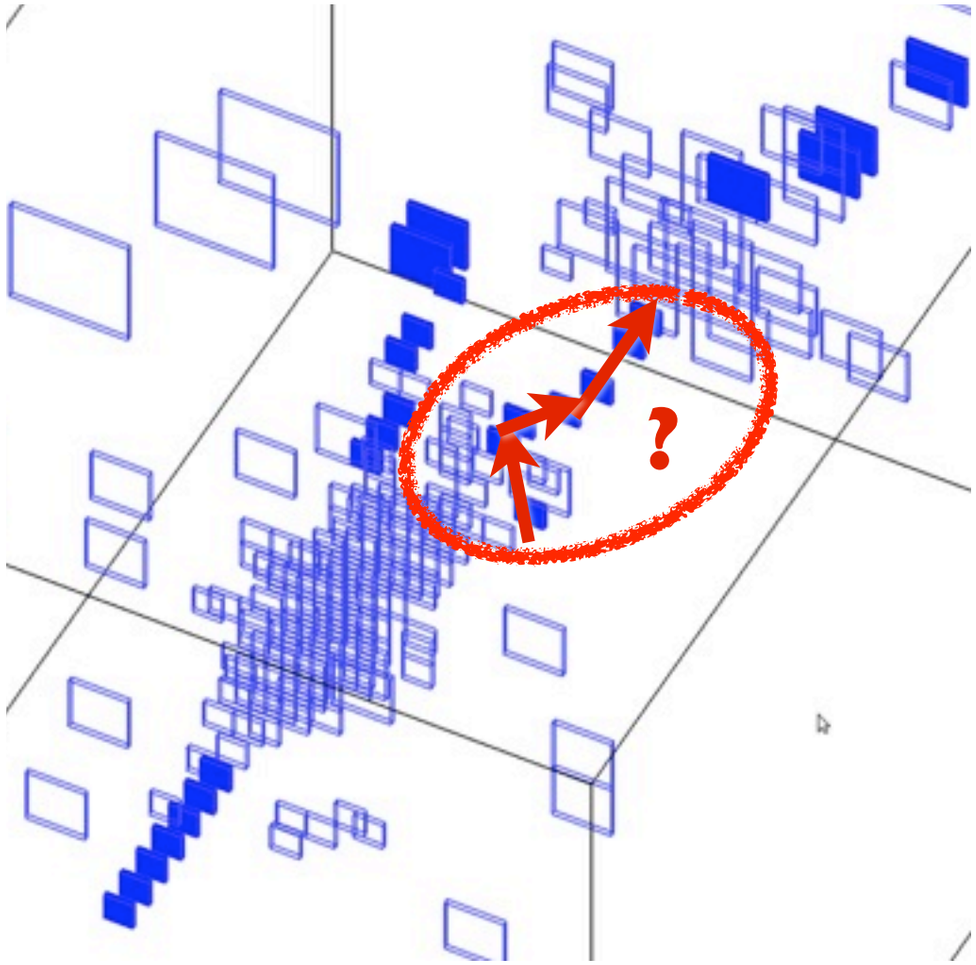
- Nearest Neigbour Algorithm
  - No usage of physical flight trajectory
  - Noise hits influence track direction
- No/Small influence on MC <=> Data comparison
- Possible solutions:
  - Track Fitting
    - not easy (e.g.: tile size is not „error“)
    - cannot fix all track errors
  - Hough Transform based filtering
    - Using Fast Hough Trafo with variable binsize

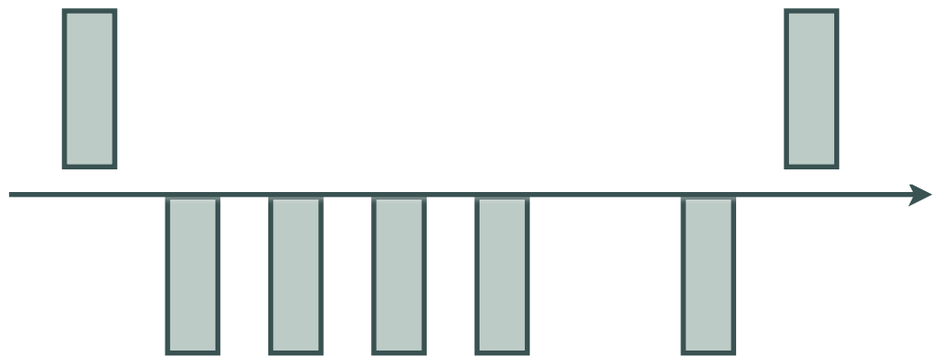




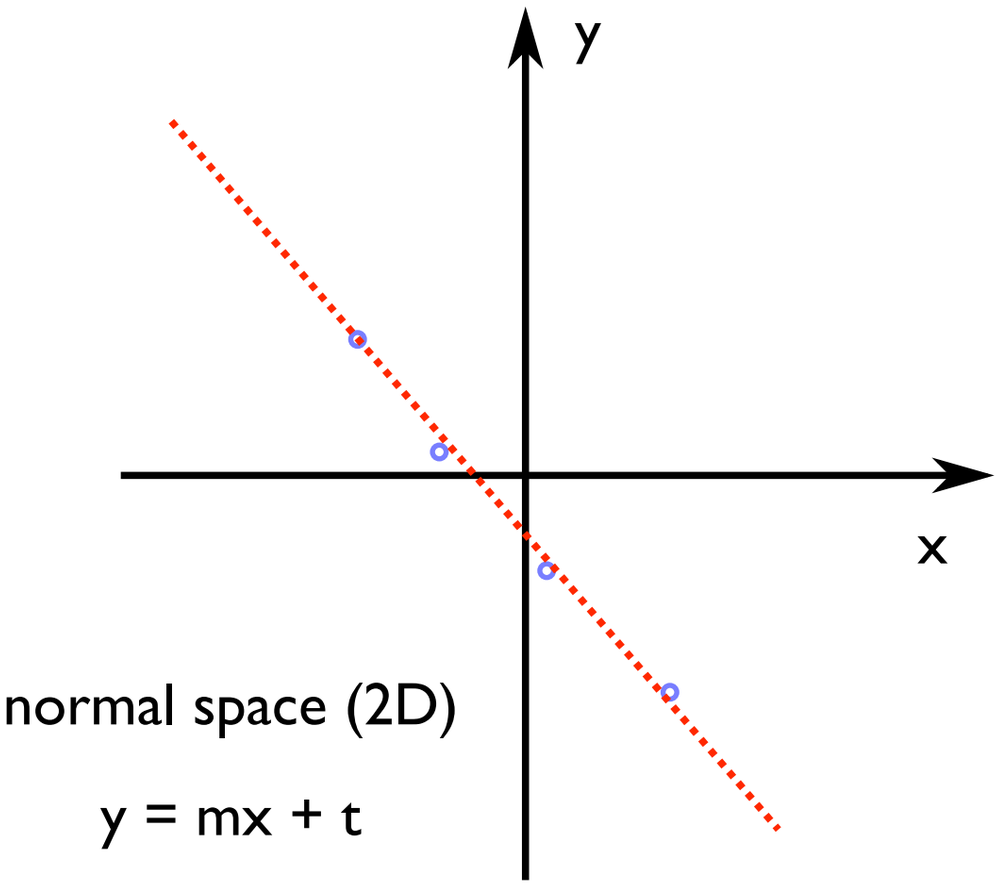
Track finding: Imperfections

- Nearest Neigbour Algorithm
  - No usage of physical flight trajectory
  - Noise hits influence track direction
- No/Small influence on MC <=> Data comparison
- Possible solutions:
  - Track Fitting
    - not easy (e.g.: tile size is not „error“)
    - cannot fix all track errors
  - Hough Transform based filtering
    - Using Fast Hough Trafo with variable binsize



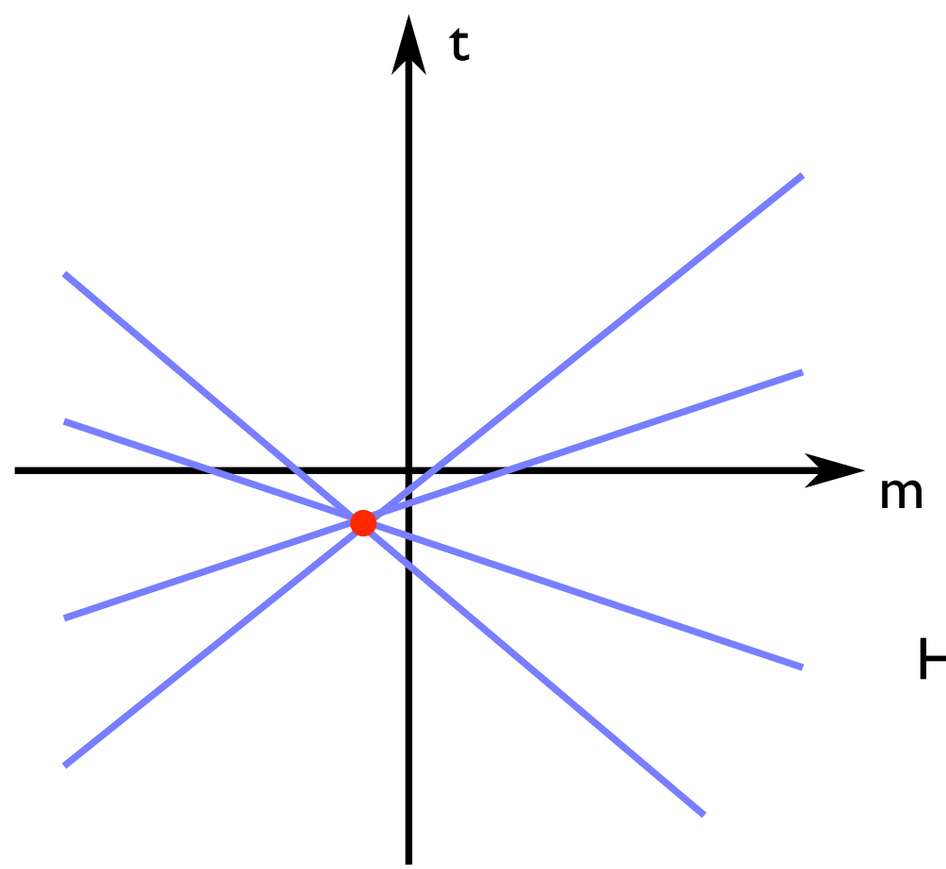


Hough transformation: Variable binsize



normal space (2D)

$y = mx + t$



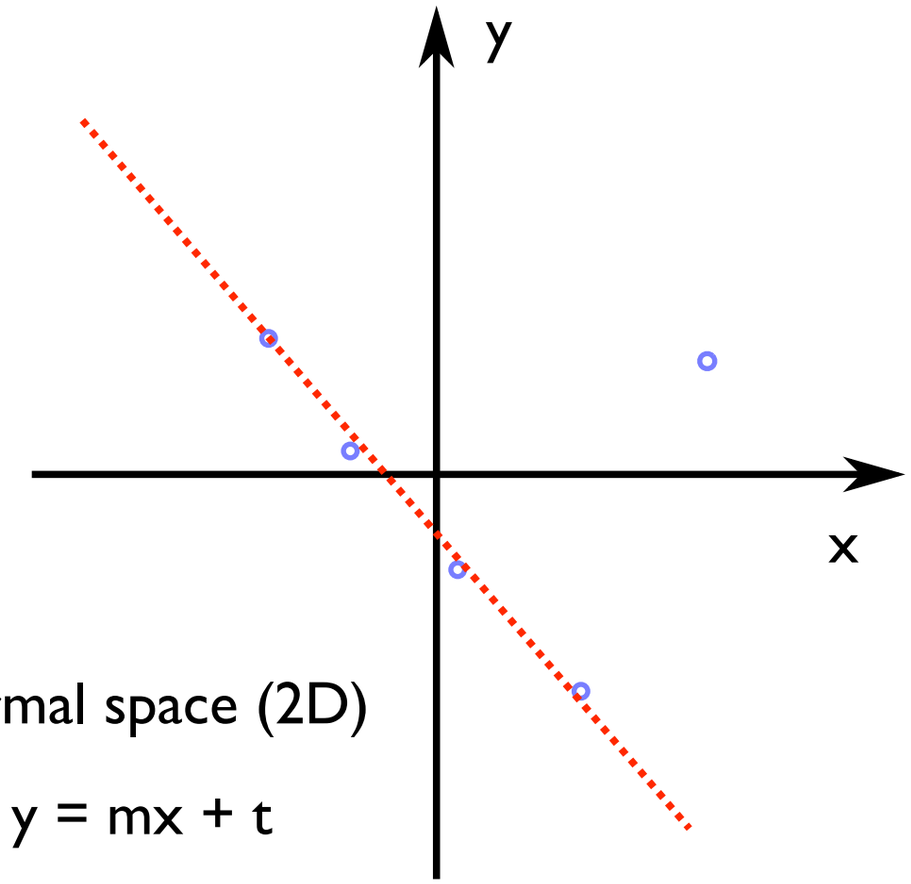
Hough Space

$t = xm - y$

- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

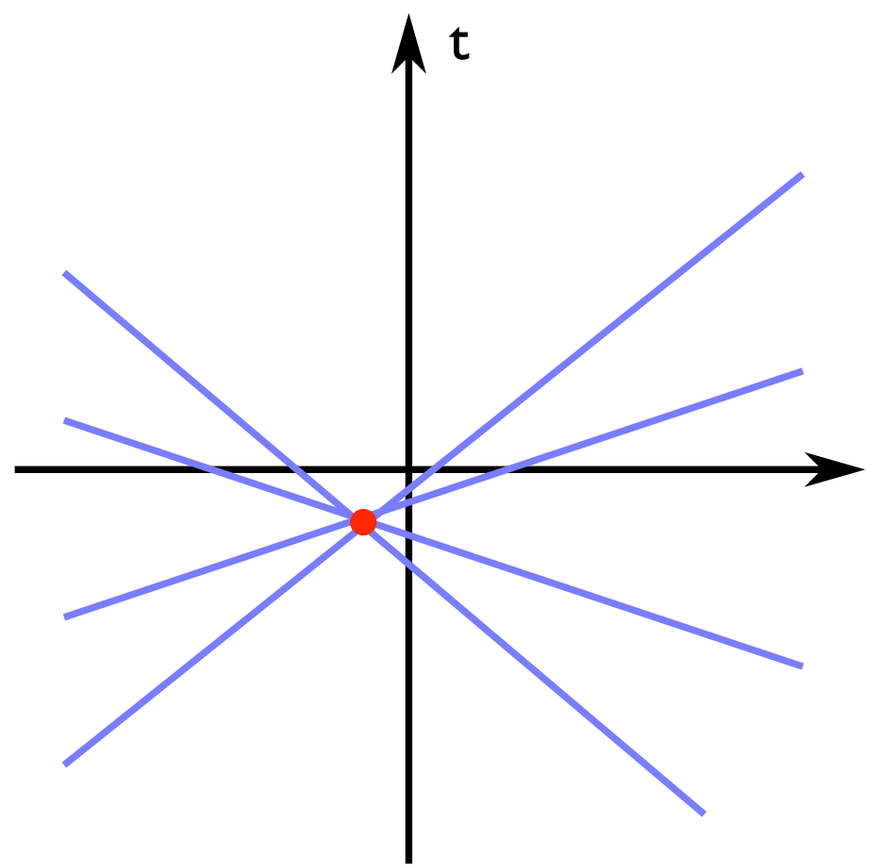
- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Variable binsize



normal space (2D)

$y = mx + t$



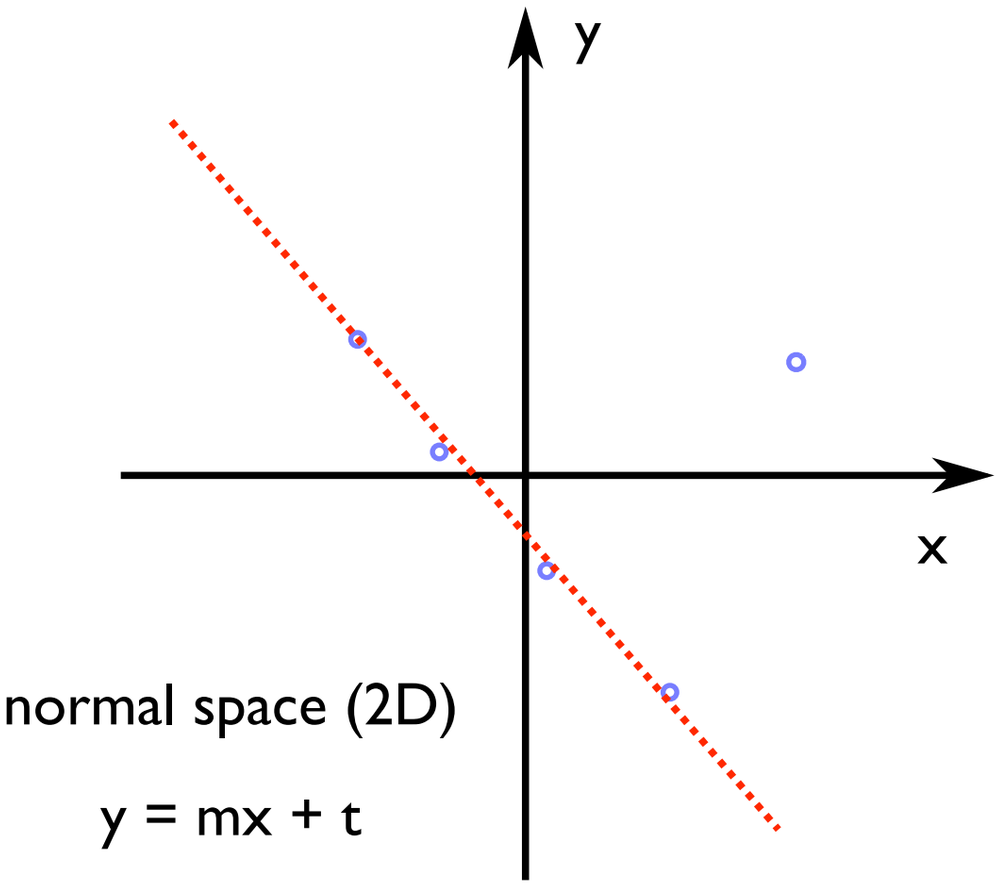
Hough Space

$t = xm - y$

- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

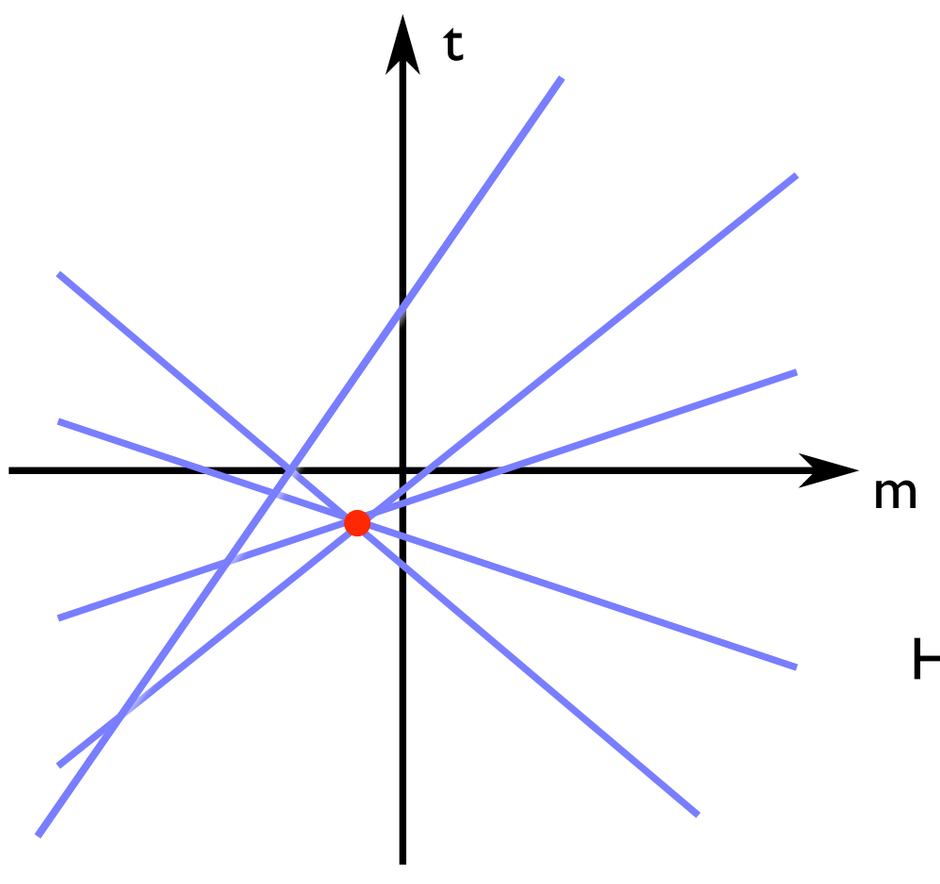
- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Variable binsize



normal space (2D)

$y = mx + t$



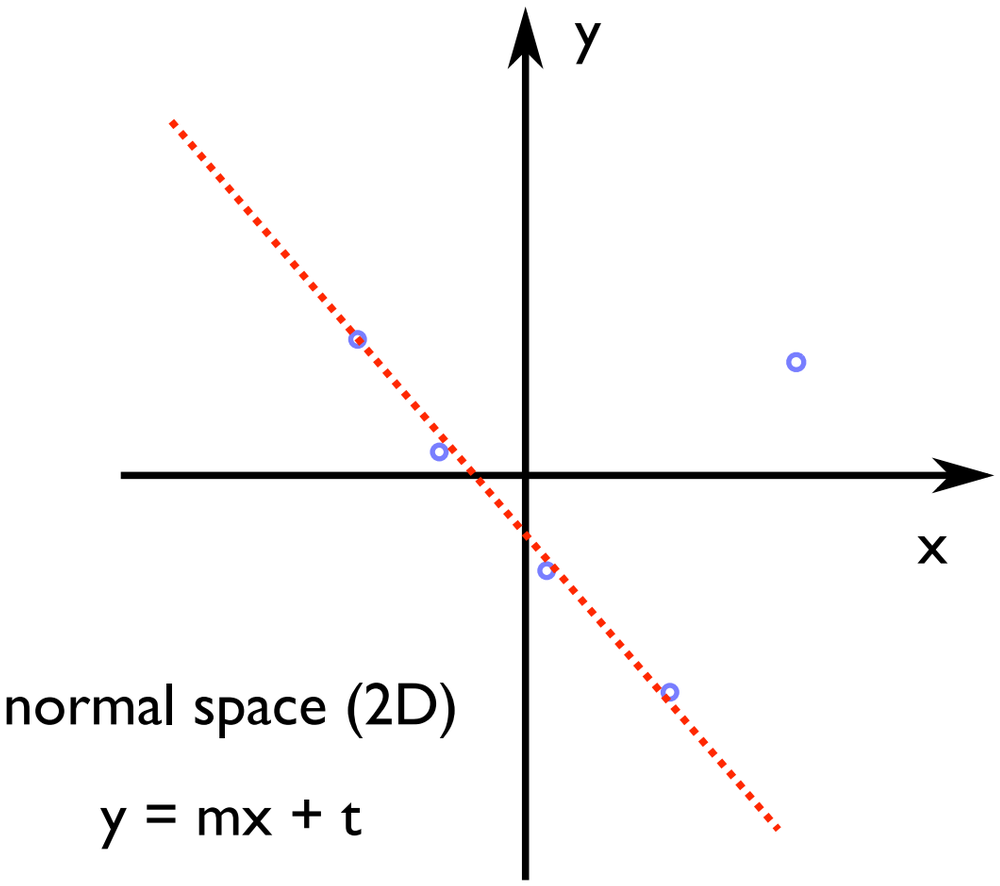
Hough Space

$t = xm - y$

- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

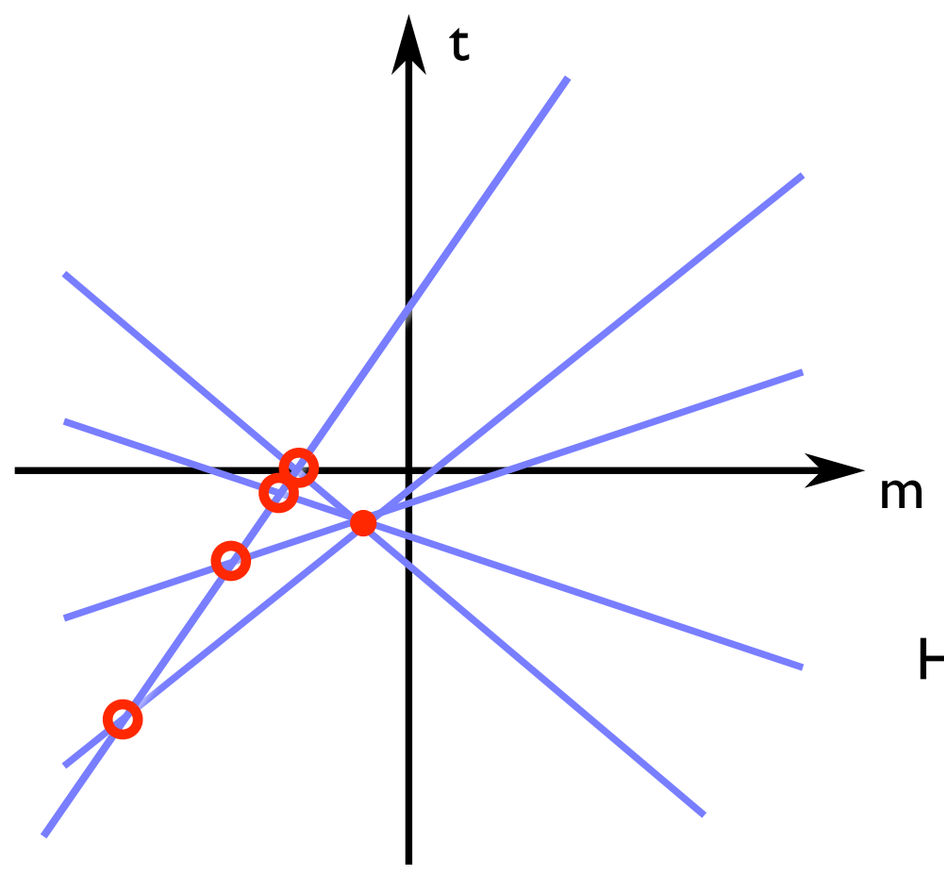
- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Variable binsize



normal space (2D)

$y = mx + t$



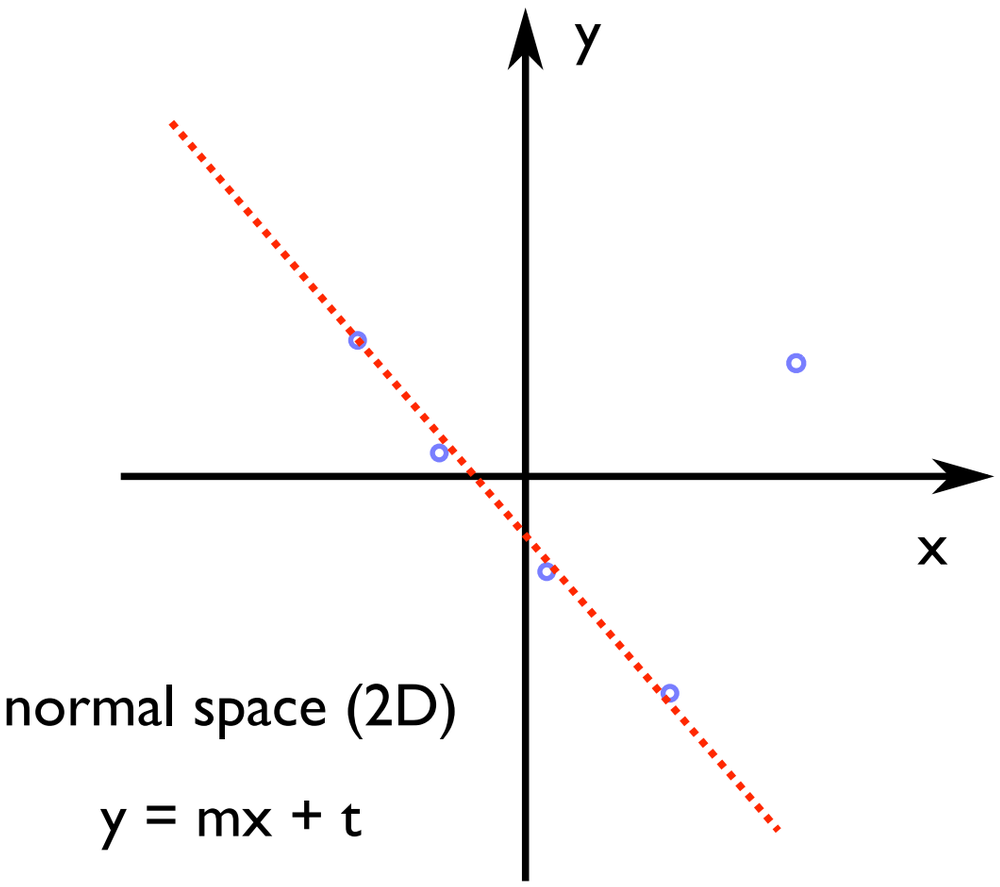
Hough Space

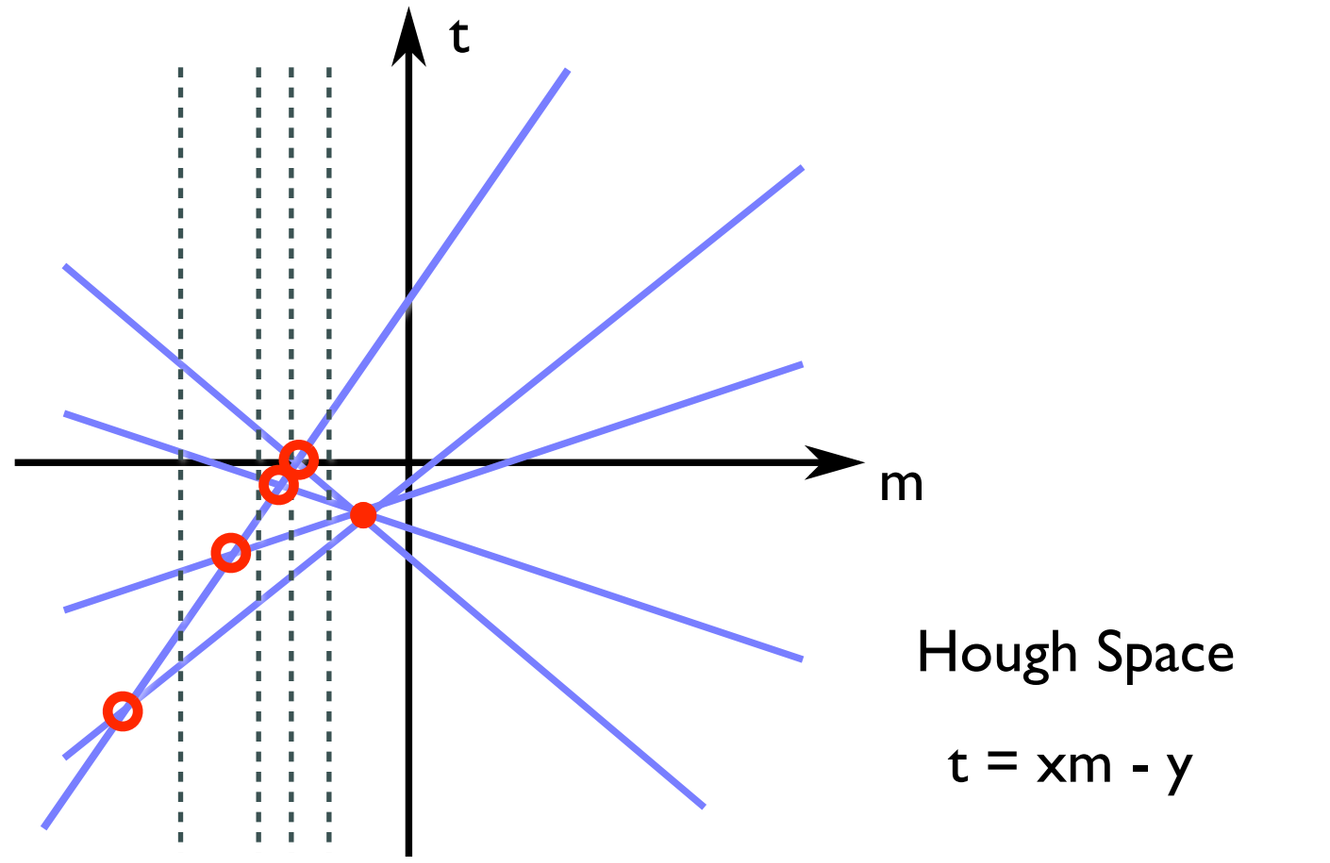
$t = xm - y$

- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Variable binsize

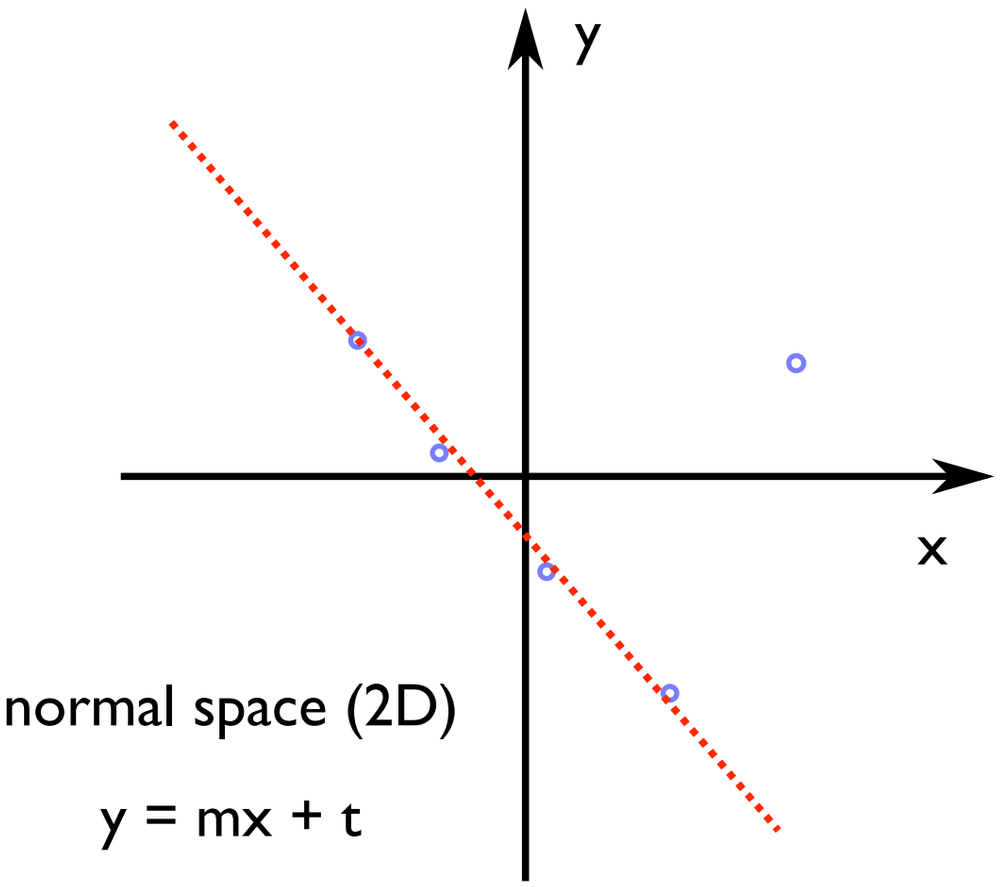




- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

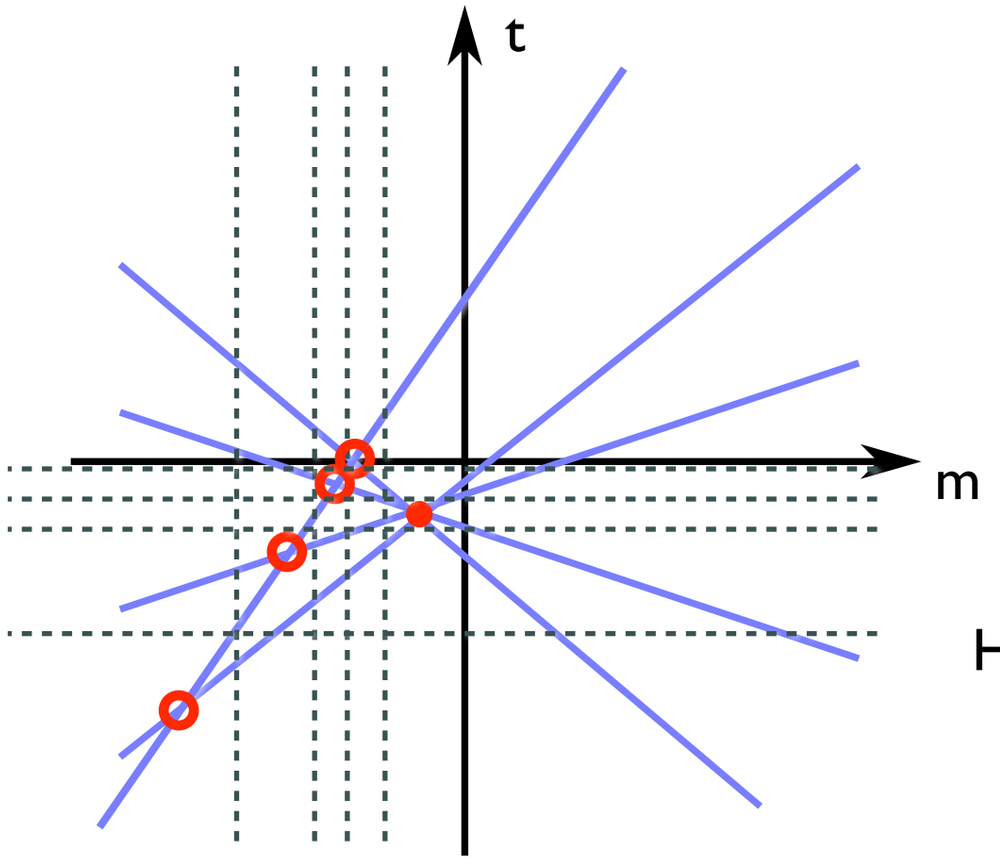
- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Variable binsize



normal space (2D)

$y = mx + t$



Hough Space

$t = xm - y$

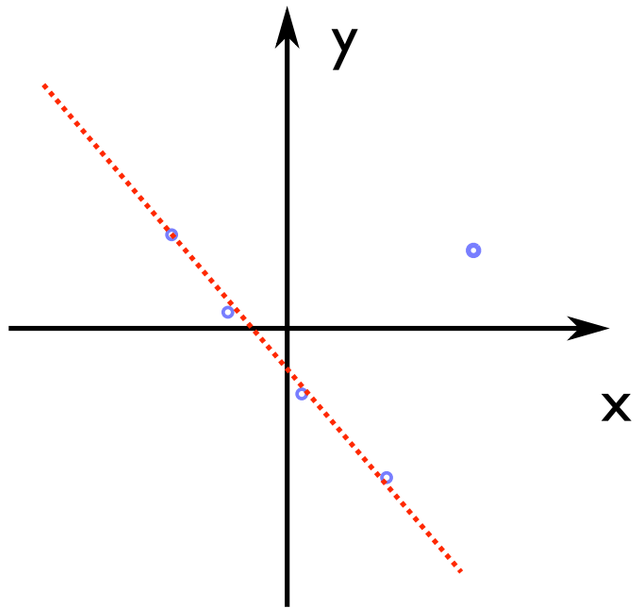
- Standard procedure
- Here: Use in 2D (x/z and y/z)
- Hough Space needs to be binned
  - RAM/CPU Usage
  - Binsize?

- Variable binsize
  - Calculate all intersections @ Hough Space
  - Take points in between two intersections as bin borders
  - Divide each bin further by 3

Hough transformation: Filtering

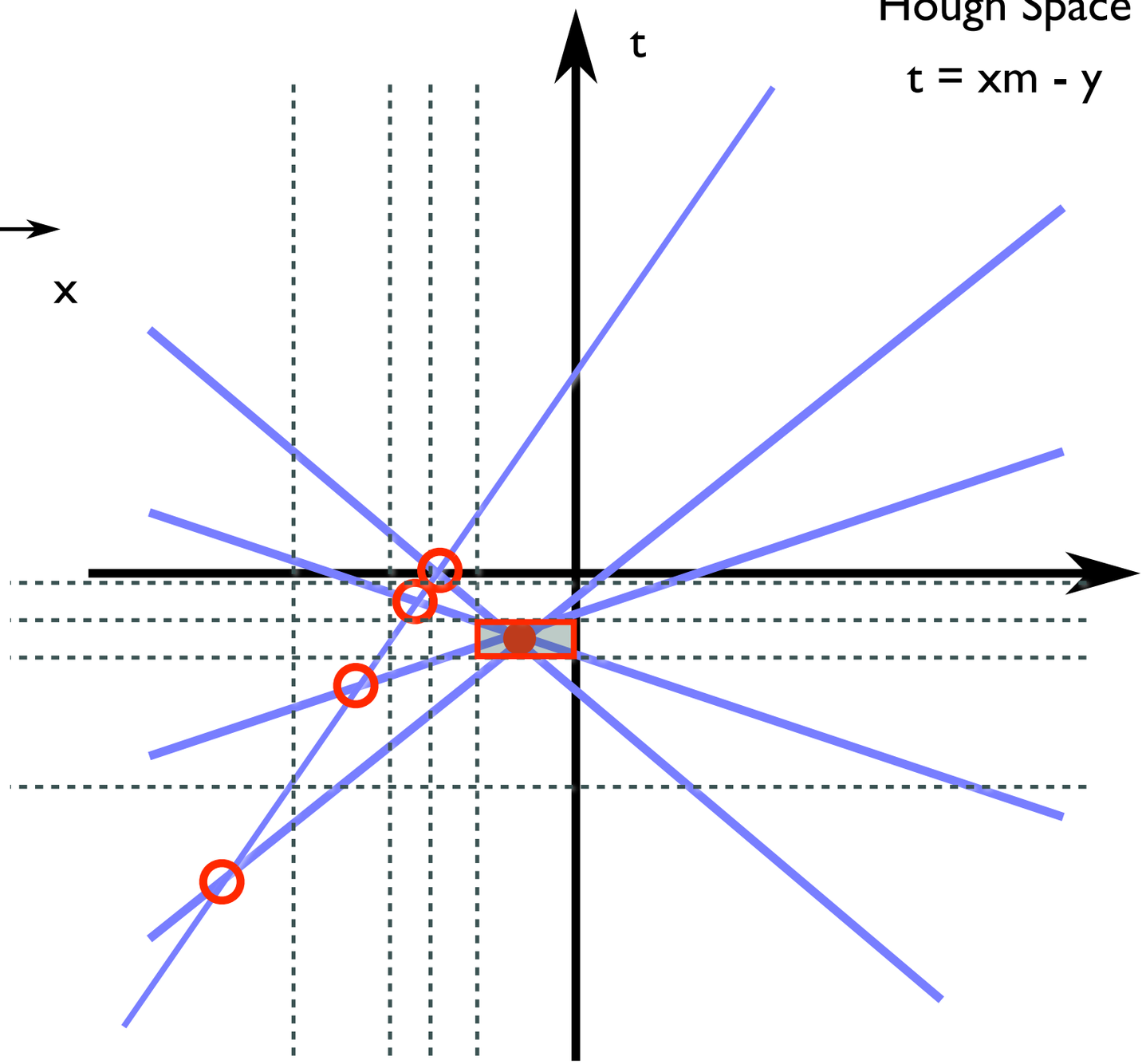
normal space (2D)

$y = mx + t$



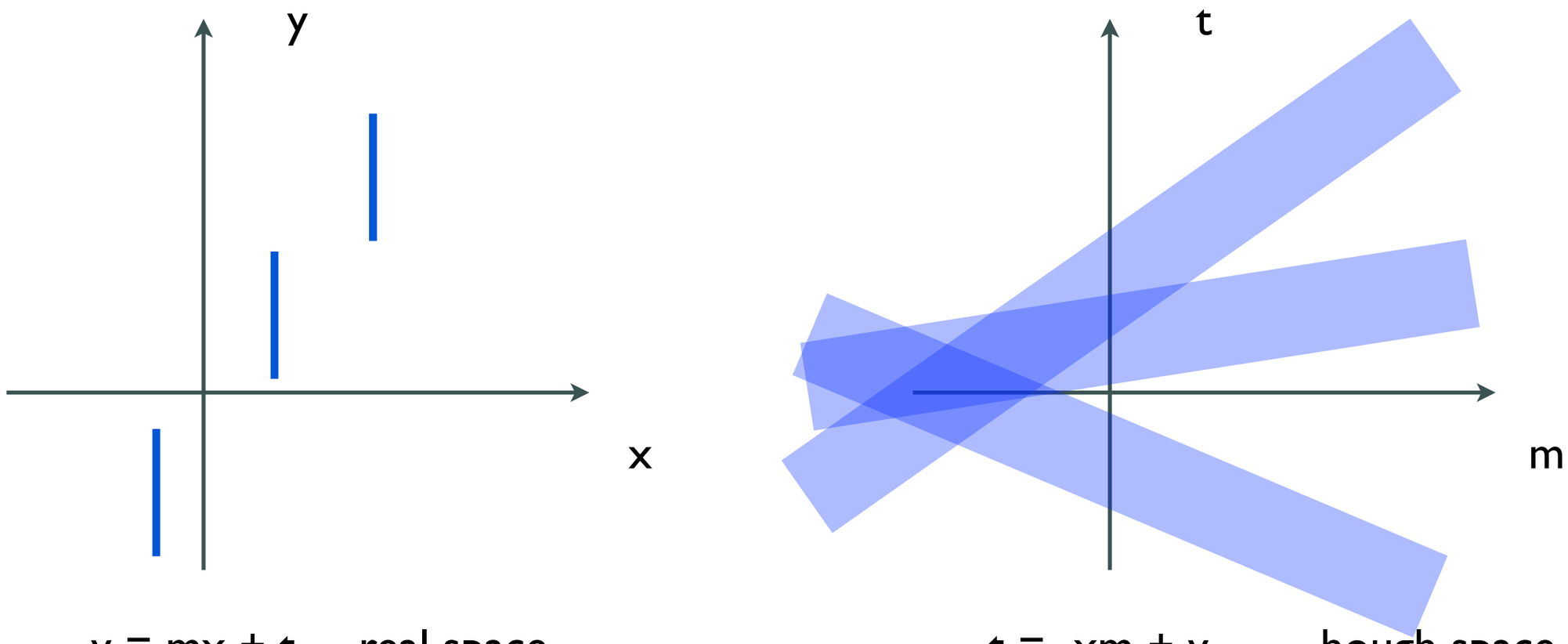
Hough Space

$t = xm - y$



- Search through the Hough Space (w/ „Divide & Conquer“)
- Identify that bin with most entries
  - All lines (hits in real space) not part of this bin ≡ noise
- HCal: cell size ≠ point like
  - use bands instead of lines in Hough Space

Hough Transformation: AHCal Hits

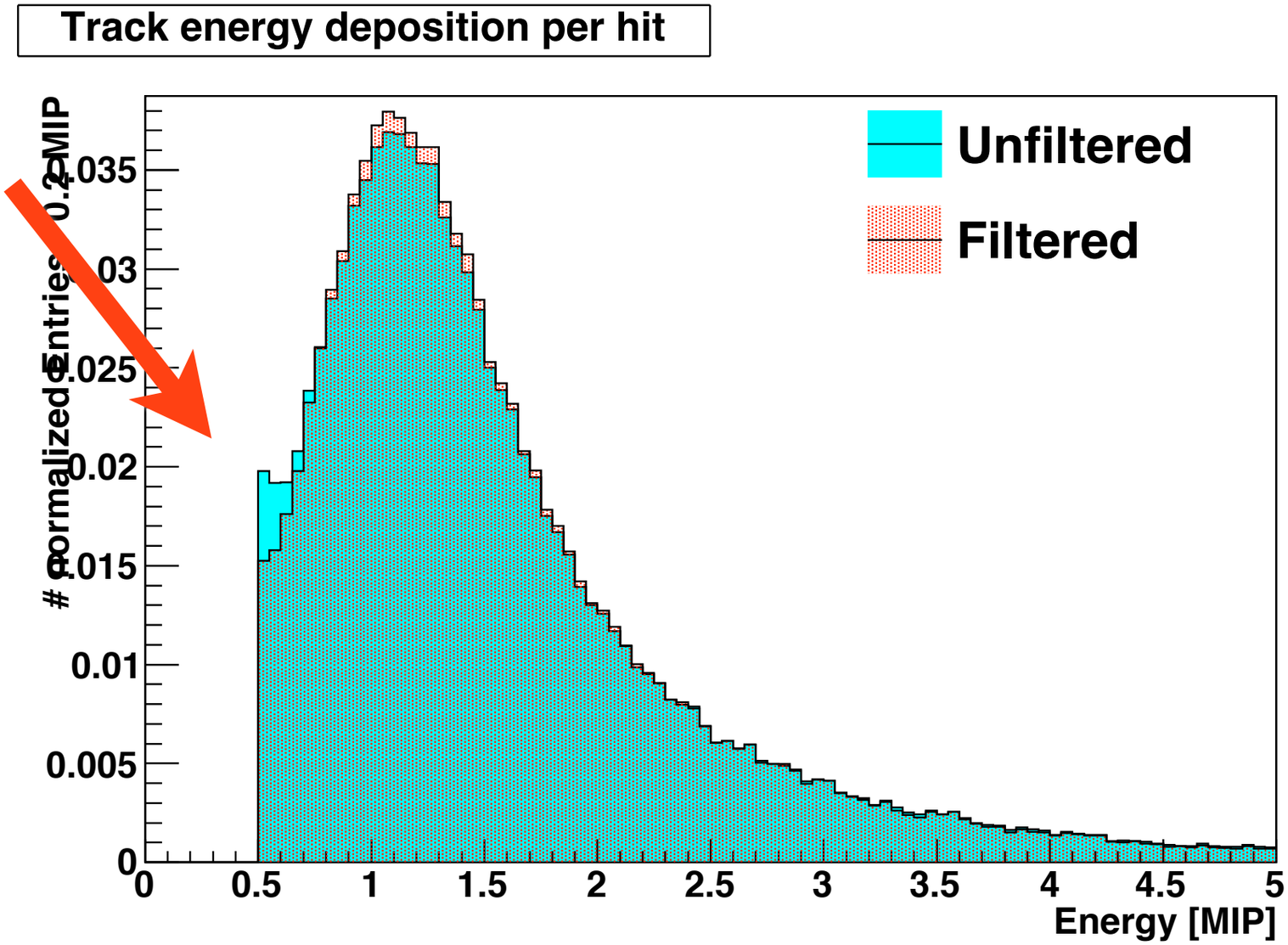


y = mx + t real space

t = -xm + y hough space

- Hit: cell size 3x3x0.5 cm^3
  - Band in hough space (neglecting thickness of 0.5cm)
- Intersection „point“ (area) difficult to calculate analytically ==> binned hough space

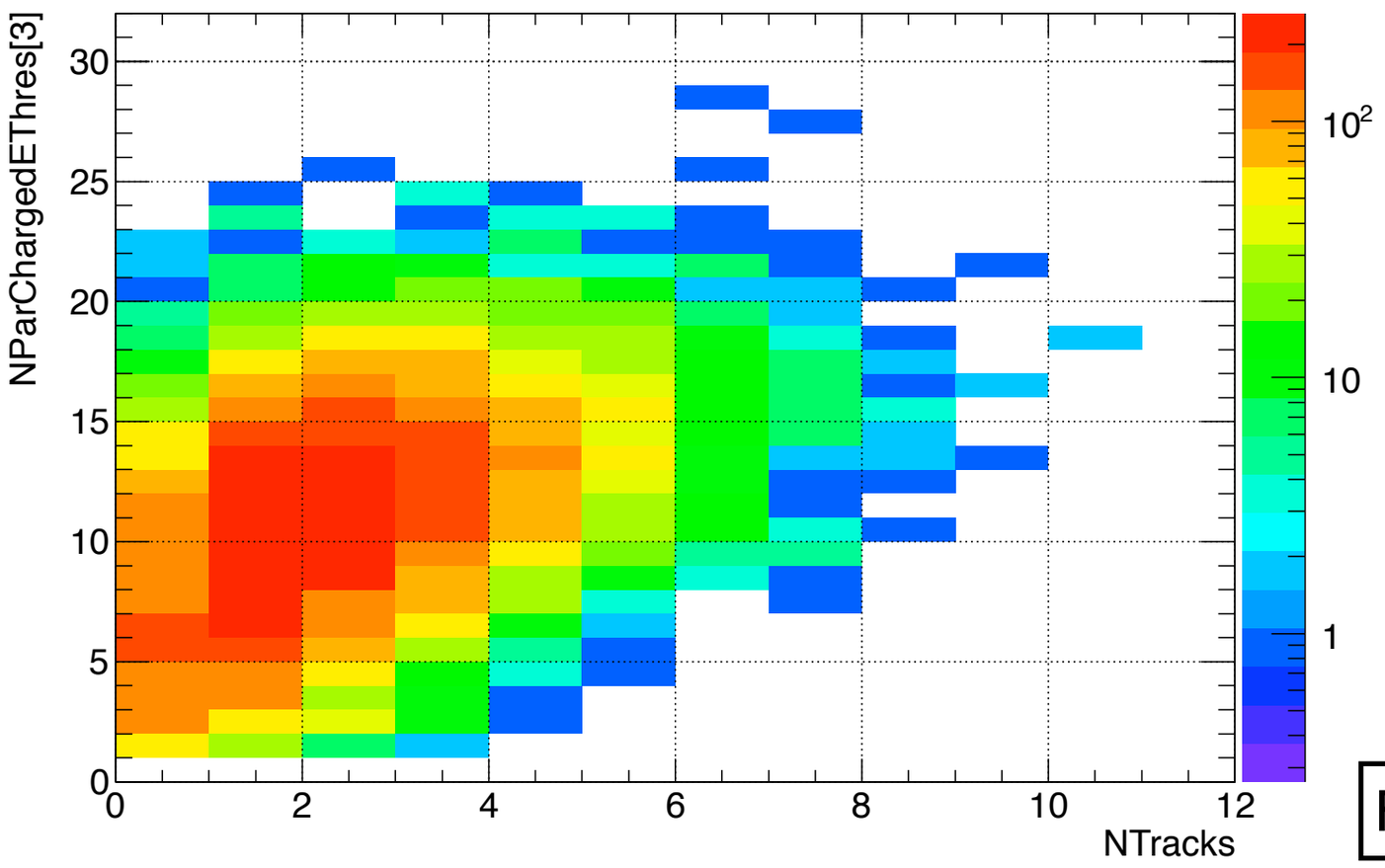
Hough Transformation: Filter results



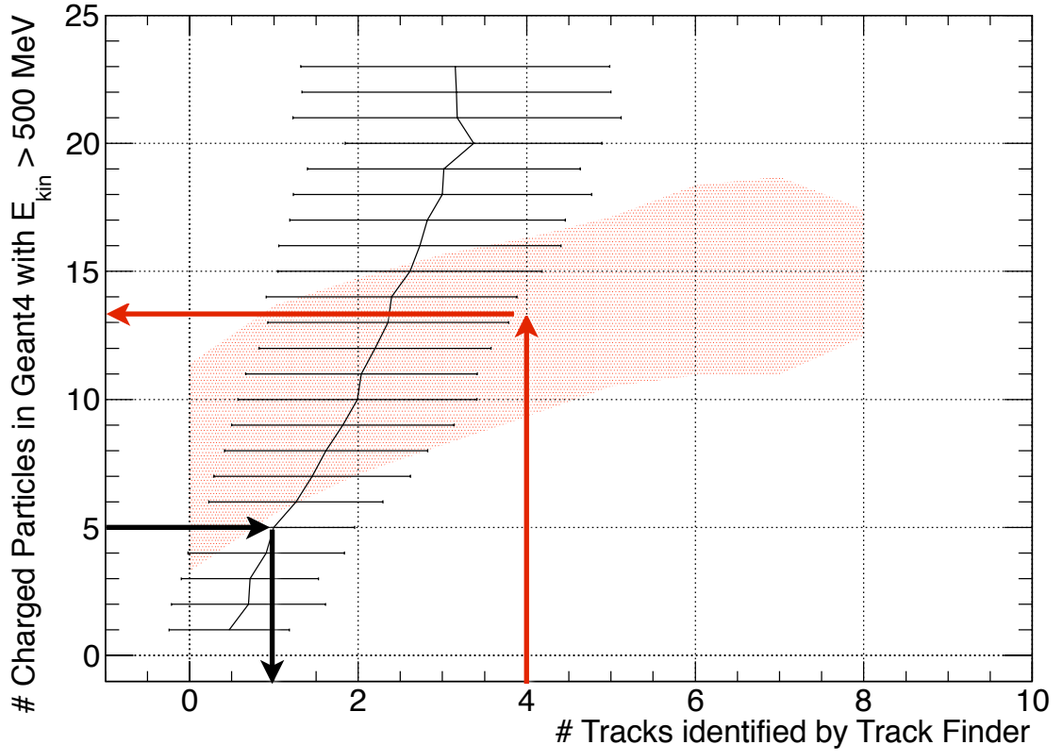
- Filterung reduces noise hits (which are main reason for unphysical tracks)
  - Here: Barely visible (low noise runs)

Track Multiplicity

- Track multiplicity <=> # Particles in hadronic shower: Correlation?
  - Mokka Hack: Convert each particle in StackingAction into MCParticle
- Here: # Tracks VS # Charged Particles with E_{kin} > 500MeV (w/o $e^{\pm}$)
  - Correlation: ~0.4 for QGSP_BERT and FTFP_BERT (LHEP: ~0.3)
  - Low multiplicity limits correlation



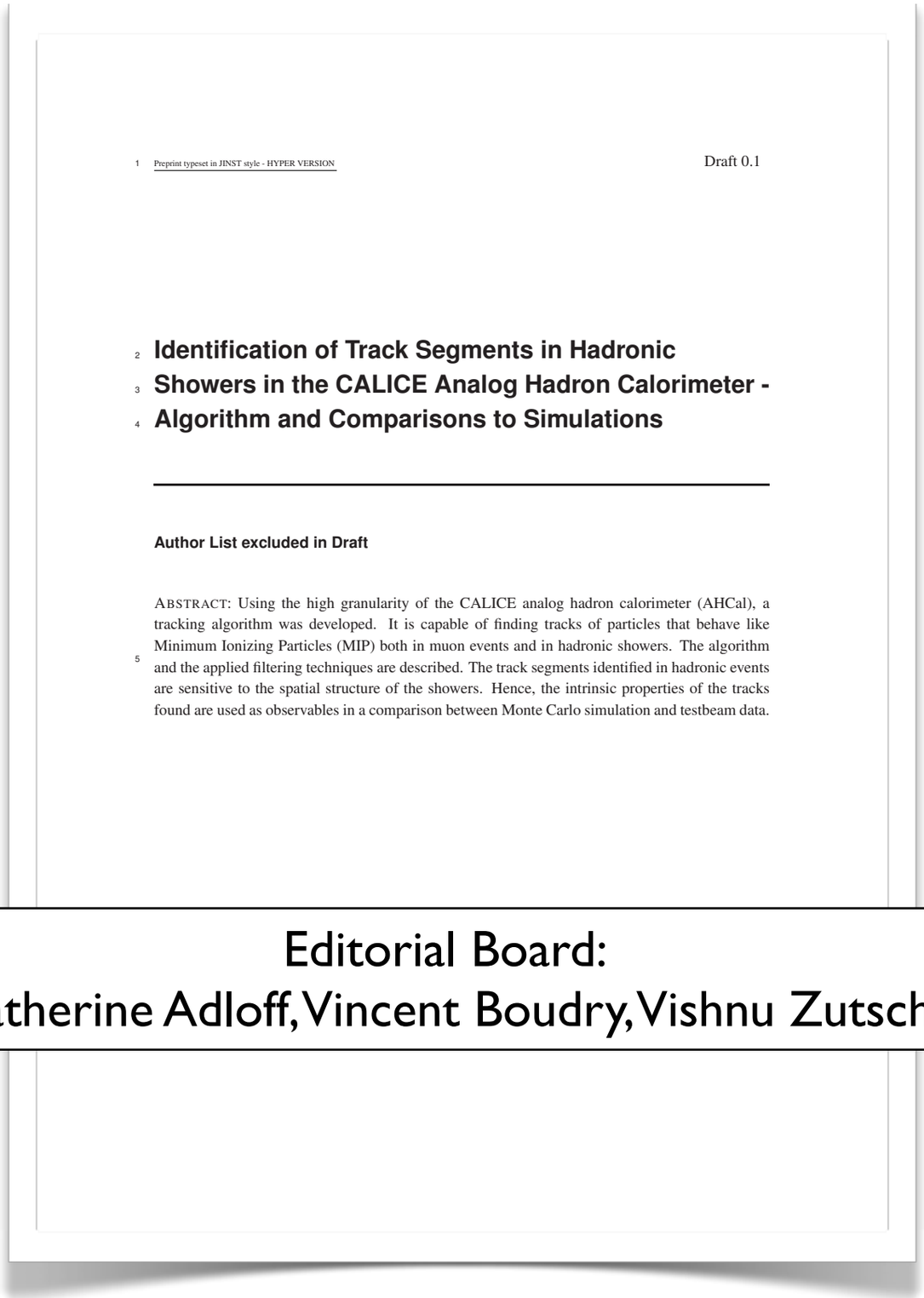
1D projection:



FTFP_BERT

Publication Plan

- Description of Algorithm + Track Filtering
- Track algorithm systematics
  - MIP Cut
  - Noise
- Data - MC comparison
  - physics lists (as requested by Geant4 team):
    - QGSP_BERT (old LHC production)
    - FTFP_BERT (new LHC production)
    - LHEP (to show how physics list evolved)
    - QGS_BIC (for systematic uncertainties)
  - Observables
    - track multiplicity
    - (typical) track length
    - track inclination



Preprint typeset in JINST style - HYPER VERSION

Draft 0.1

Identification of Track Segments in Hadronic Showers in the CALICE Analog Hadron Calorimeter - Algorithm and Comparisons to Simulations

Author List excluded in Draft

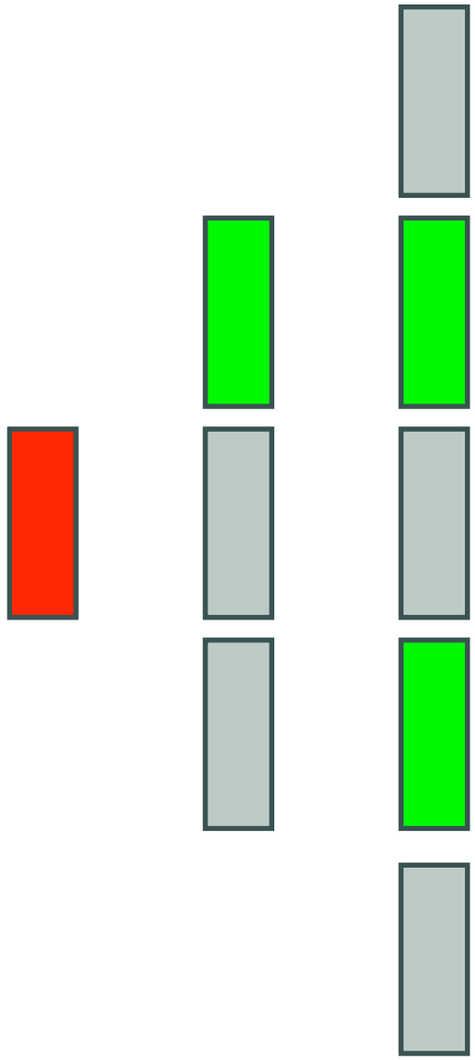
ABSTRACT: Using the high granularity of the CALICE analog hadron calorimeter (AHCal), a tracking algorithm was developed. It is capable of finding tracks of particles that behave like Minimum Ionizing Particles (MIP) both in muon events and in hadronic showers. The algorithm and the applied filtering techniques are described. The track segments identified in hadronic events are sensitive to the spatial structure of the showers. Hence, the intrinsic properties of the tracks found are used as observables in a comparison between Monte Carlo simulation and testbeam data.

Editorial Board:
Catherine Adloff, Vincent Boudry, Vishnu Zutschi

Summary

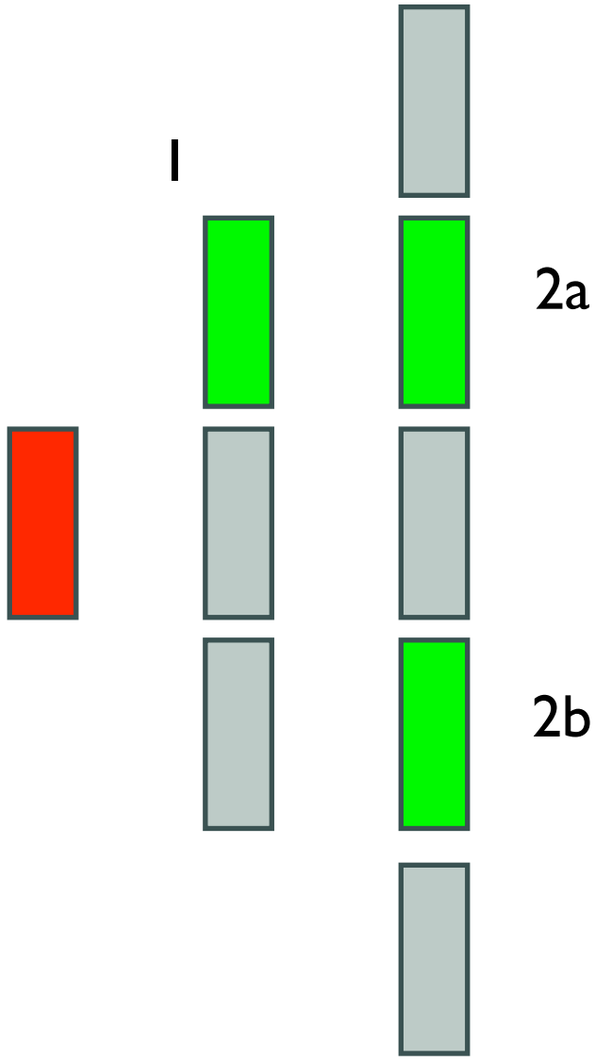
- Tracking algorithm is working
- New Hough Transformation based filter to reject outliers
- Correlation: track mulitplicity <=> # charged particles
- First draft almost complete

Backup: Tracking Algorithm



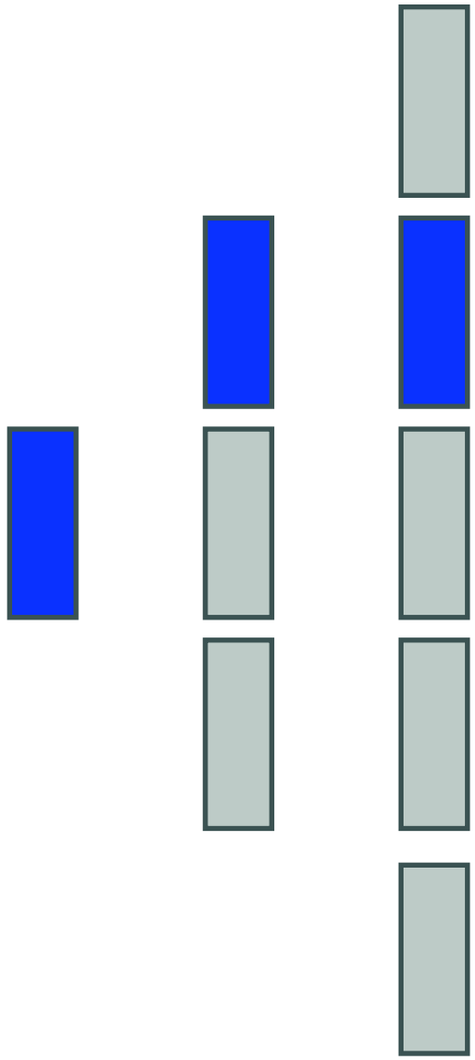
- For each isolated hit as track start:
  - Collect isolated hits in the consecutive 2 layers
  - Sort them by distance to track start
  - Search for tracks with each of these as new track start point
    - Avoid double counting of hits
    - (Here: Hit 1 will use Hit 2a in its track, hence there is no possibility to start an independent track from Hit 2a)
- Merge with longest track

Backup: Tracking Algorithm



- For each isolated hit as track start:
  - Collect isolated hits in the consecutive 2 layers
  - Sort them by distance to track start
  - Search for tracks with each of these as new track start point
    - Avoid double counting of hits
    - (Here: Hit 1 will use Hit 2a in its track, hence there is no possibility to start an independent track from Hit 2a)
  - Merge with longest track

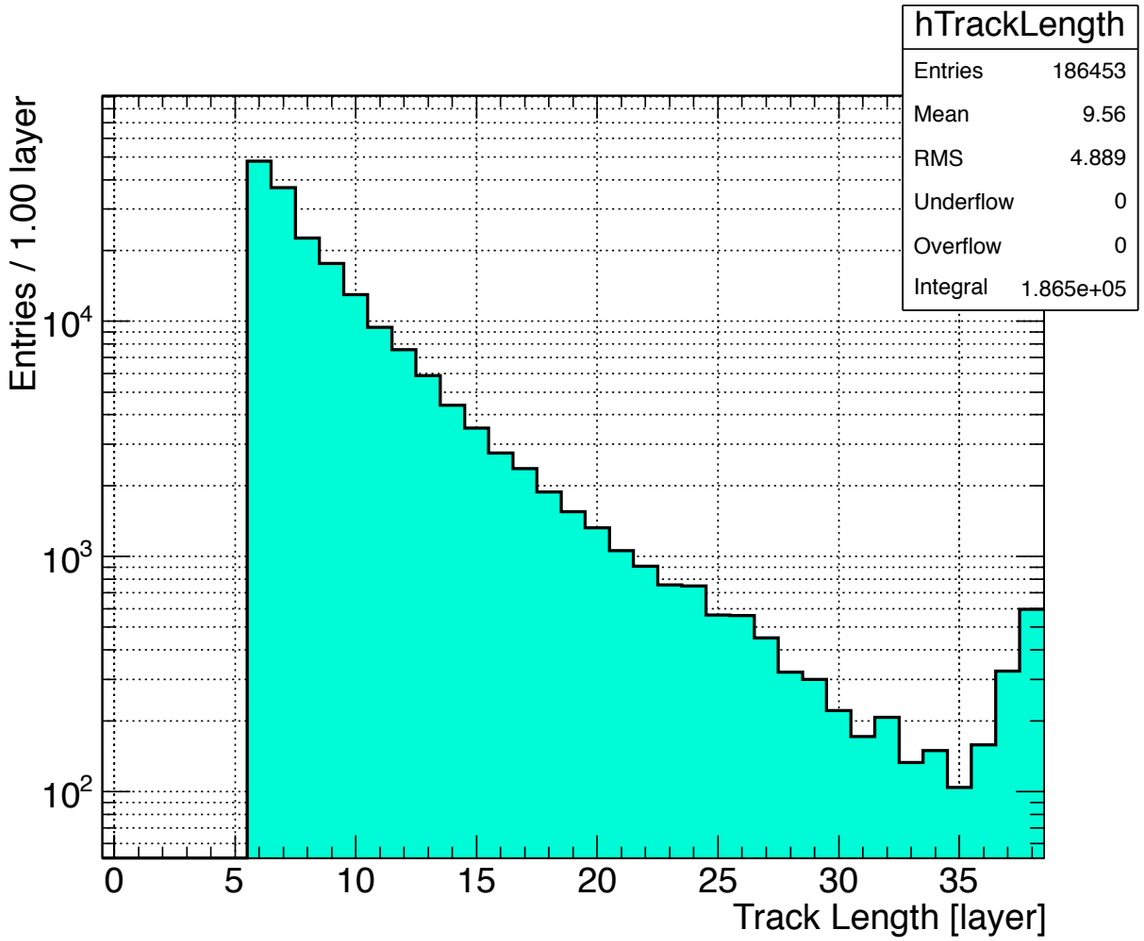
Backup: Tracking Algorithm

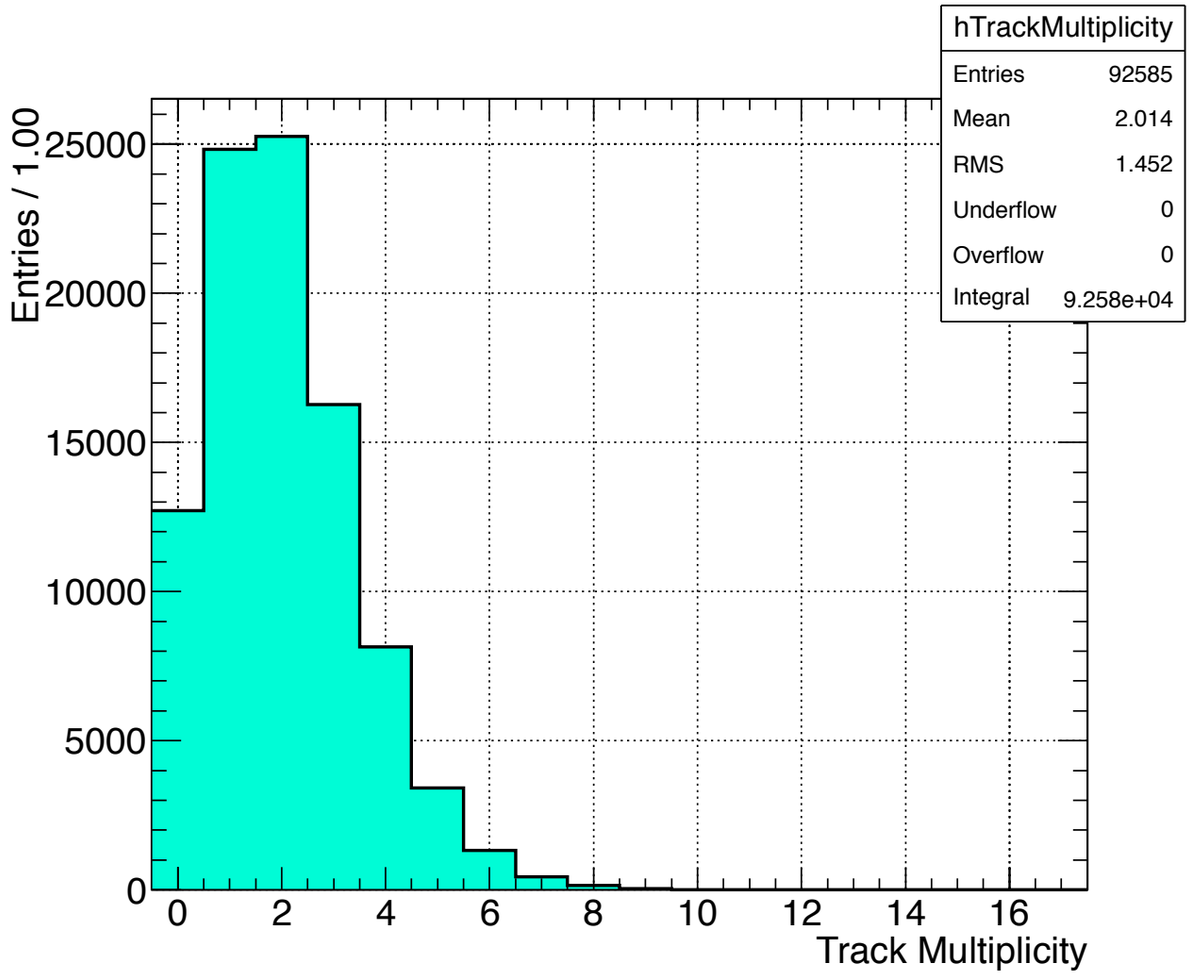


- For each isolated hit as track start:
  - Collect isolated hits in the consecutive 2 layers
  - Sort them by distance to track start
  - Search for tracks with each of these as new track start point
    - Avoid double counting of hits
    - (Here: Hit 1 will use Hit 2a in its track, hence there is no possibility to start an independent track from Hit 2a)
- Merge with longest track

Track multiplicity / Track length

- For Run 330325
  - 25 GeV pi-
- On average: 2.01 tracks / evt
  - Old tracker (different 25 GeV run): 1.6 tracks / evt





- Exponentially decreasing tracklength

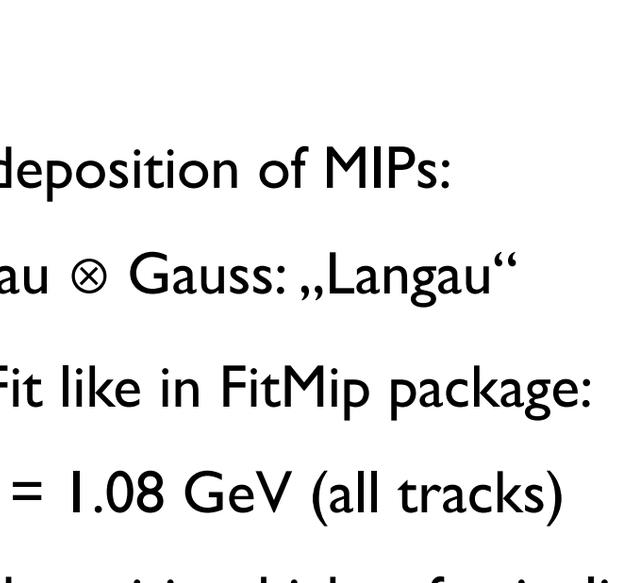
→ hadronic interaction length λ_0

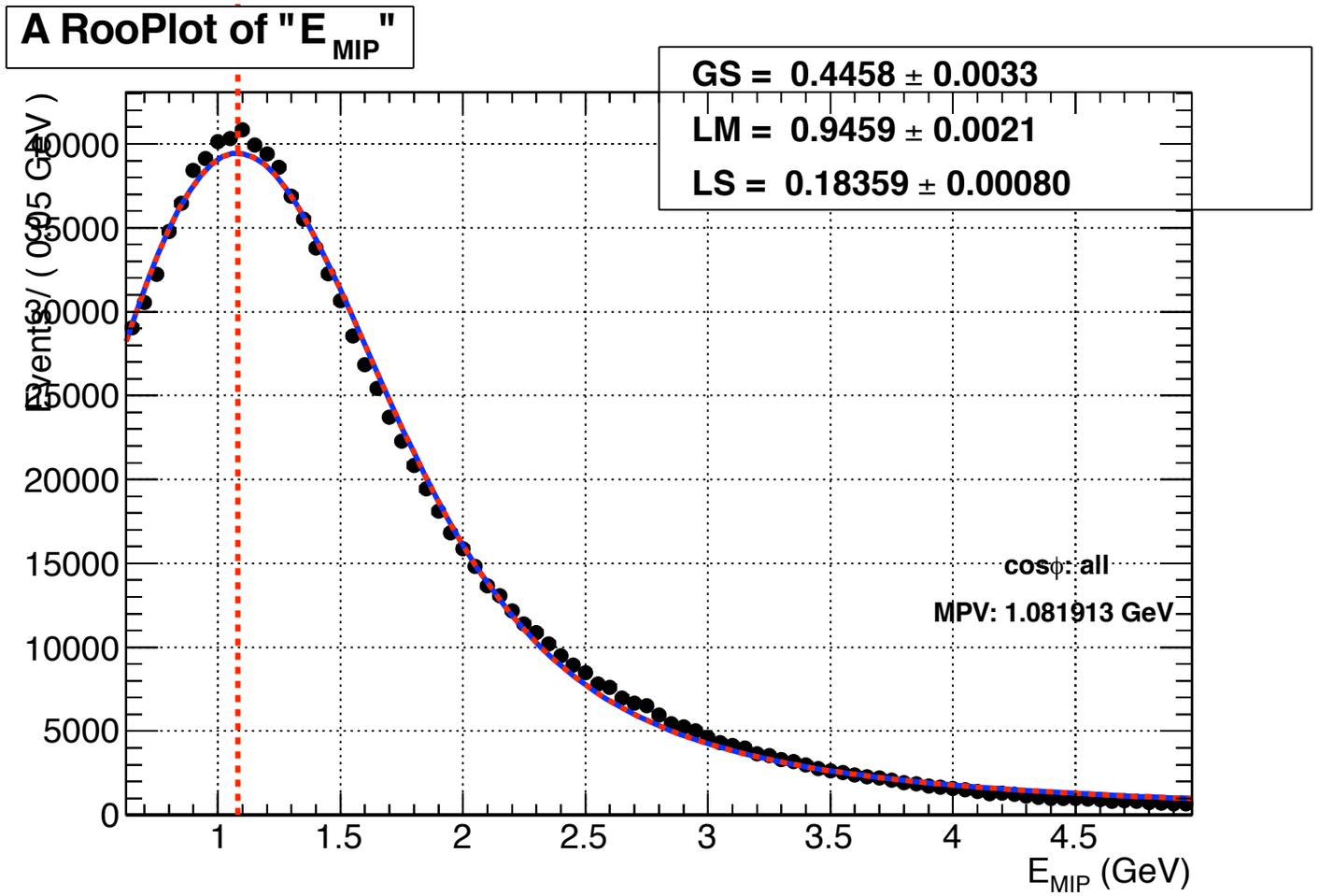
- Quick estimation of λ_0 (straight, primary tracks):

$$\lambda_0 = 8.1 \text{ layers}$$
$$\lambda_{0,PDG} = 8.88 \text{ layers}$$

Track segments by MIPs: Langau

- Energy deposition of MIPs:
  - Landau ⊗ Gauss: „Langau“
- Similar Fit like in FitMip package:
  - MPV = 1.08 GeV (all tracks)
- Energy deposition higher for inclined tracks
  - MPV = 0.99 GeV (straight tracks)

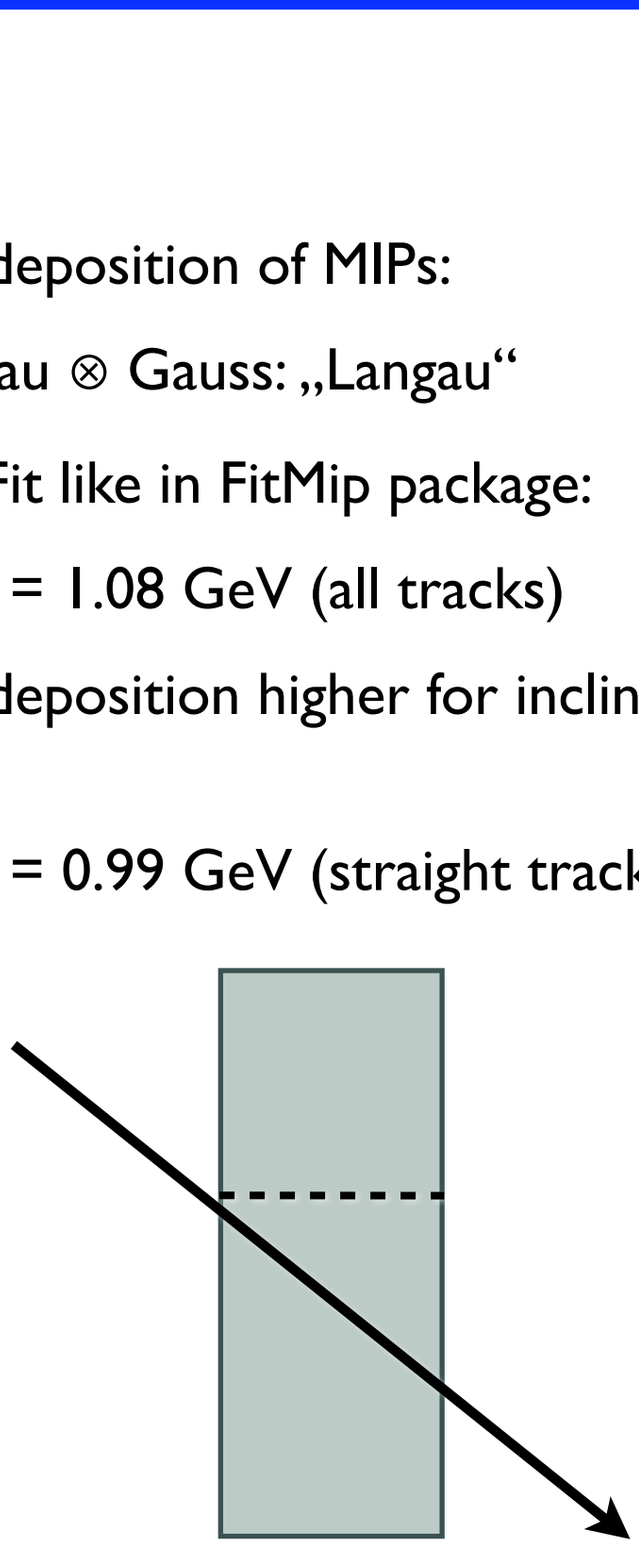


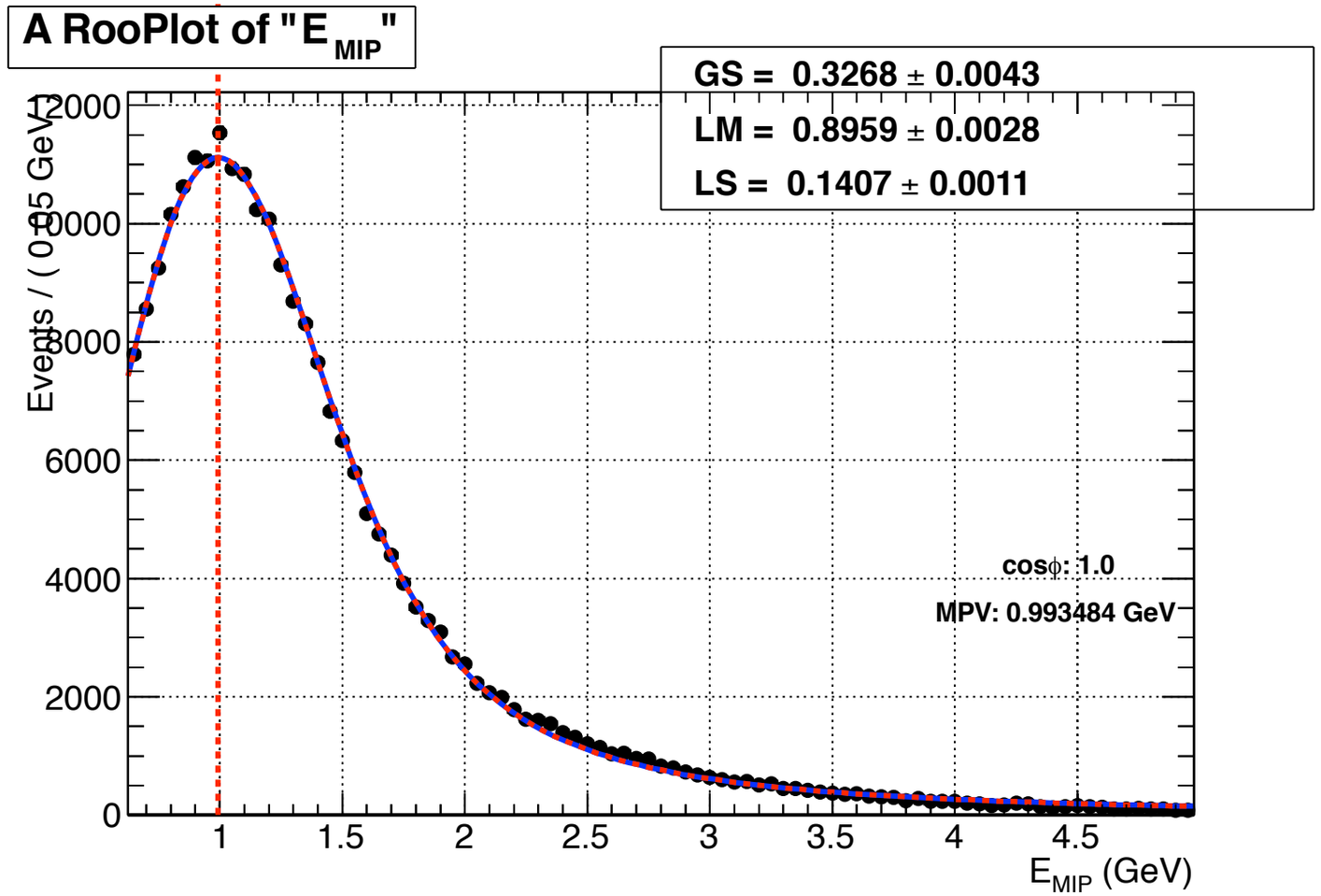


Run 331333: 60 GeV Pion

Track segments by MIPs: Langau

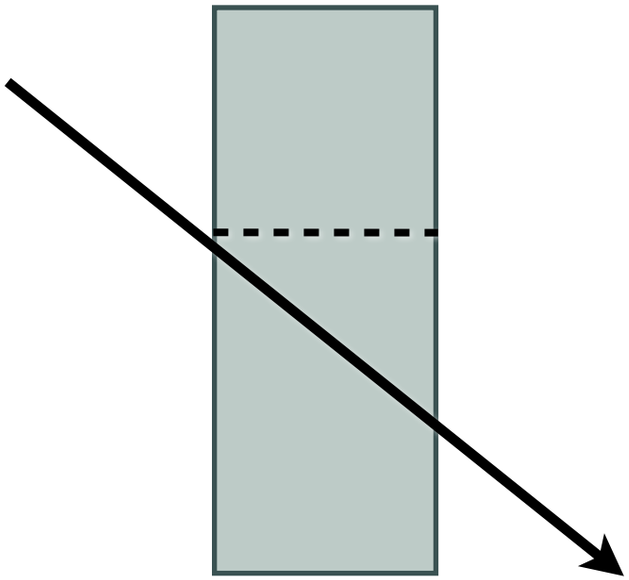
- Energy deposition of MIPs:
  - Landau ⊗ Gauss: „Langau“
- Similar Fit like in FitMip package:
  - MPV = 1.08 GeV (all tracks)
- Energy deposition higher for inclined tracks
  - MPV = 0.99 GeV (straight tracks)



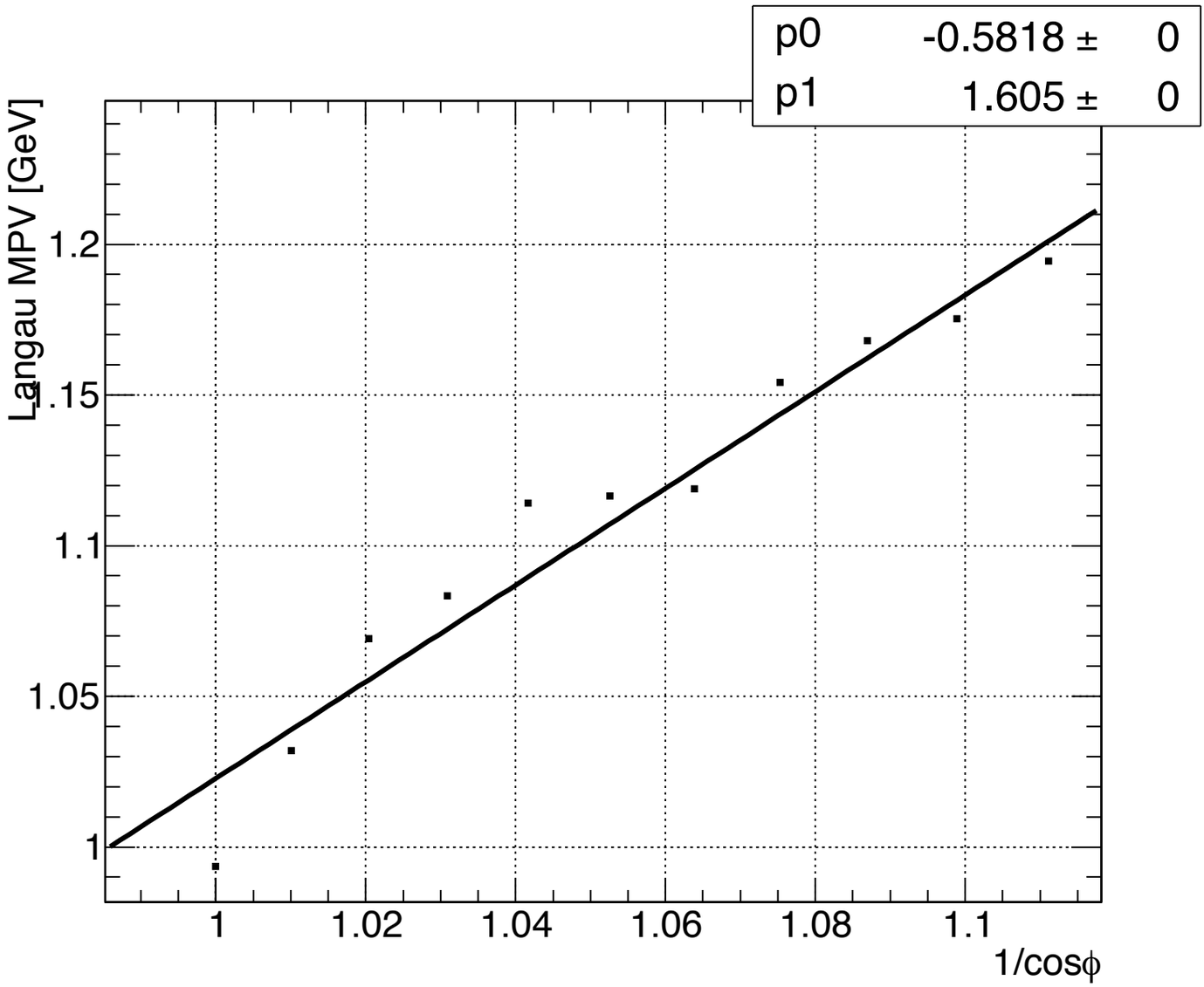


Run 331333: 60 GeV Pion

Langau MPV: Track angle dependence

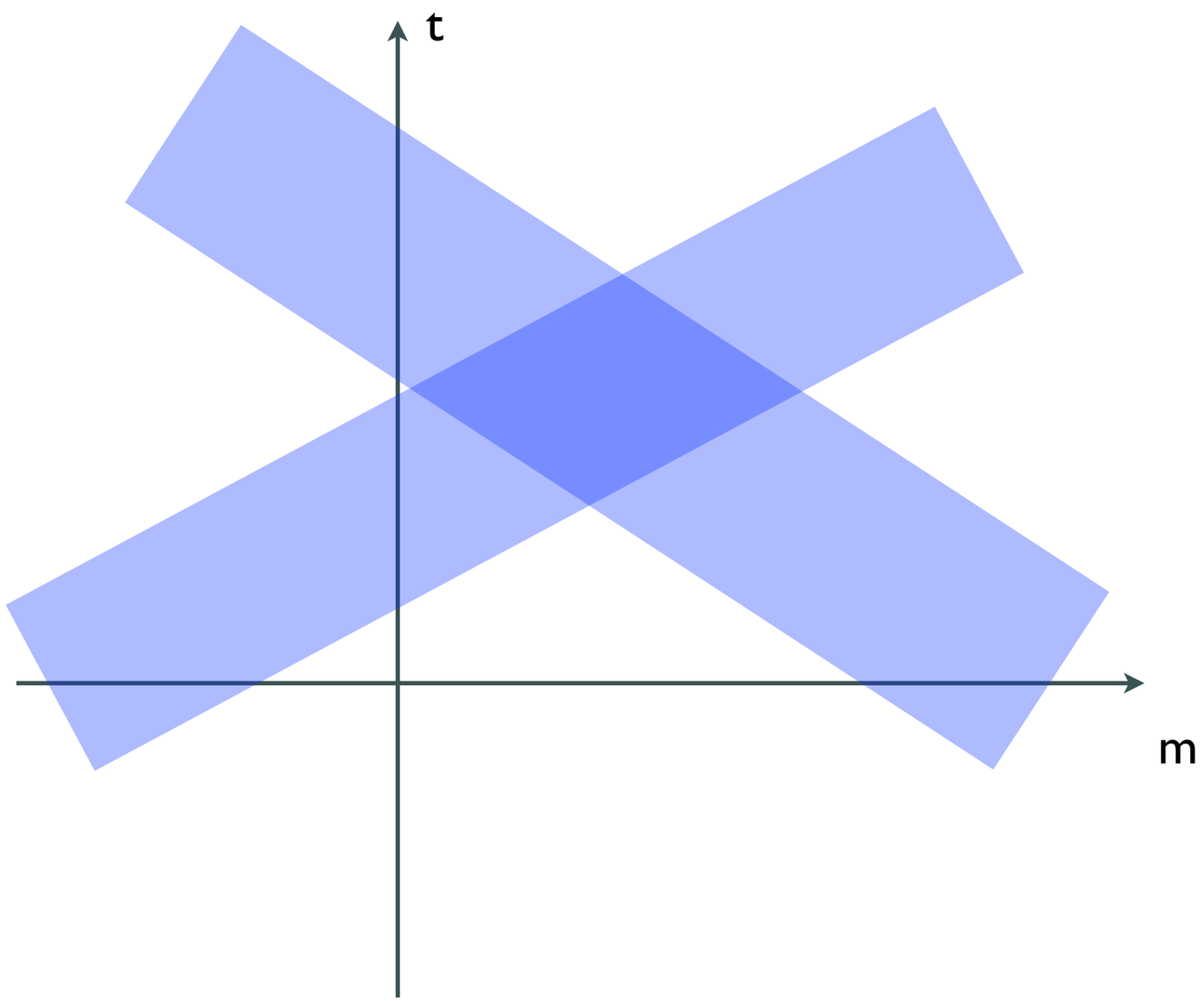


- Longer distance @ inclined tracks
  - higher E_{dep}
- expected: $E_{dep} \propto 1/\cos\varphi$



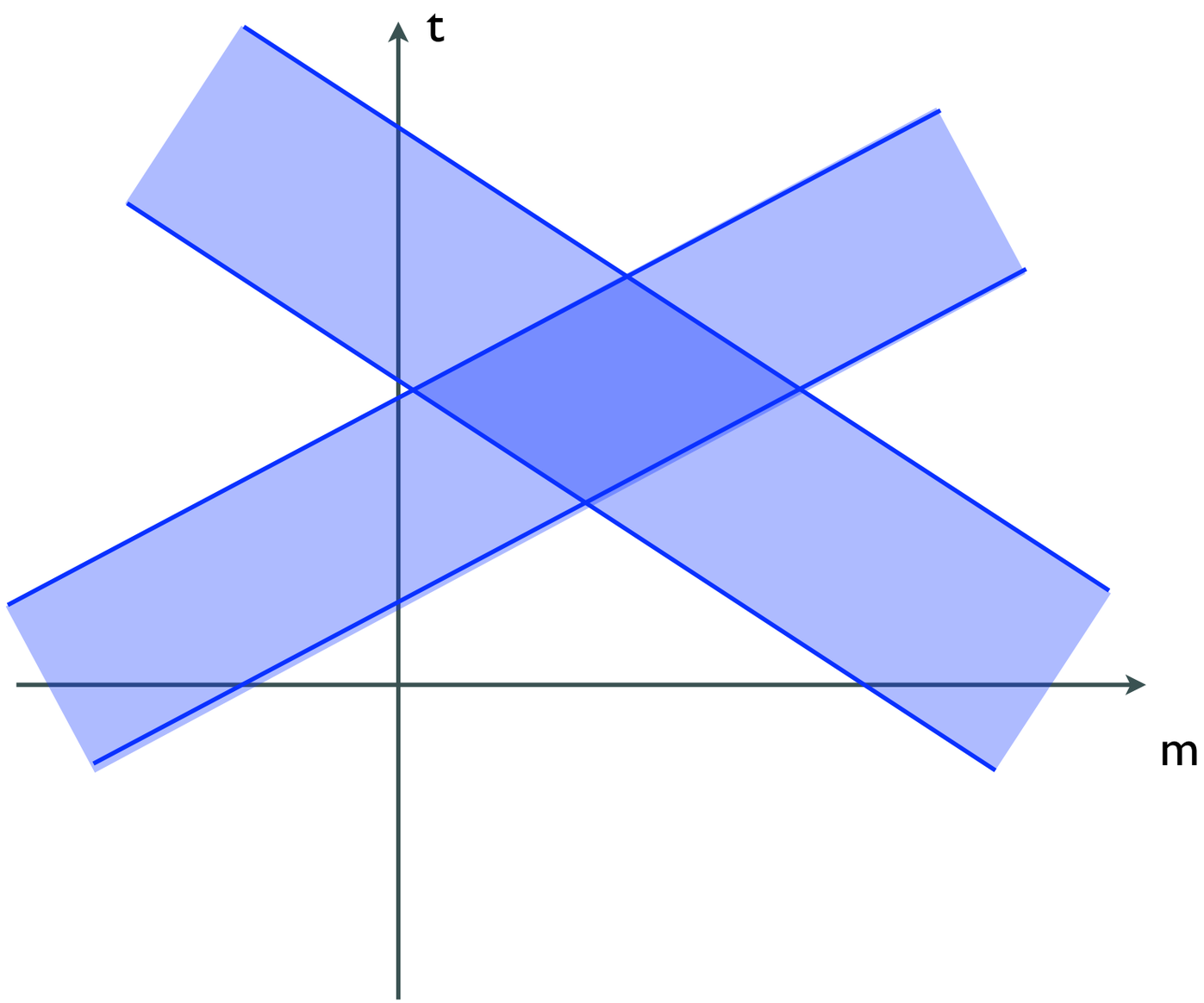
Hough Transformation: Variable binsize

- Use lines limiting the bands
- Calculate intersections between all limiting lines
- Bin border = mid between 2 intersections in m (t)
- Fill bin if band traverses bin



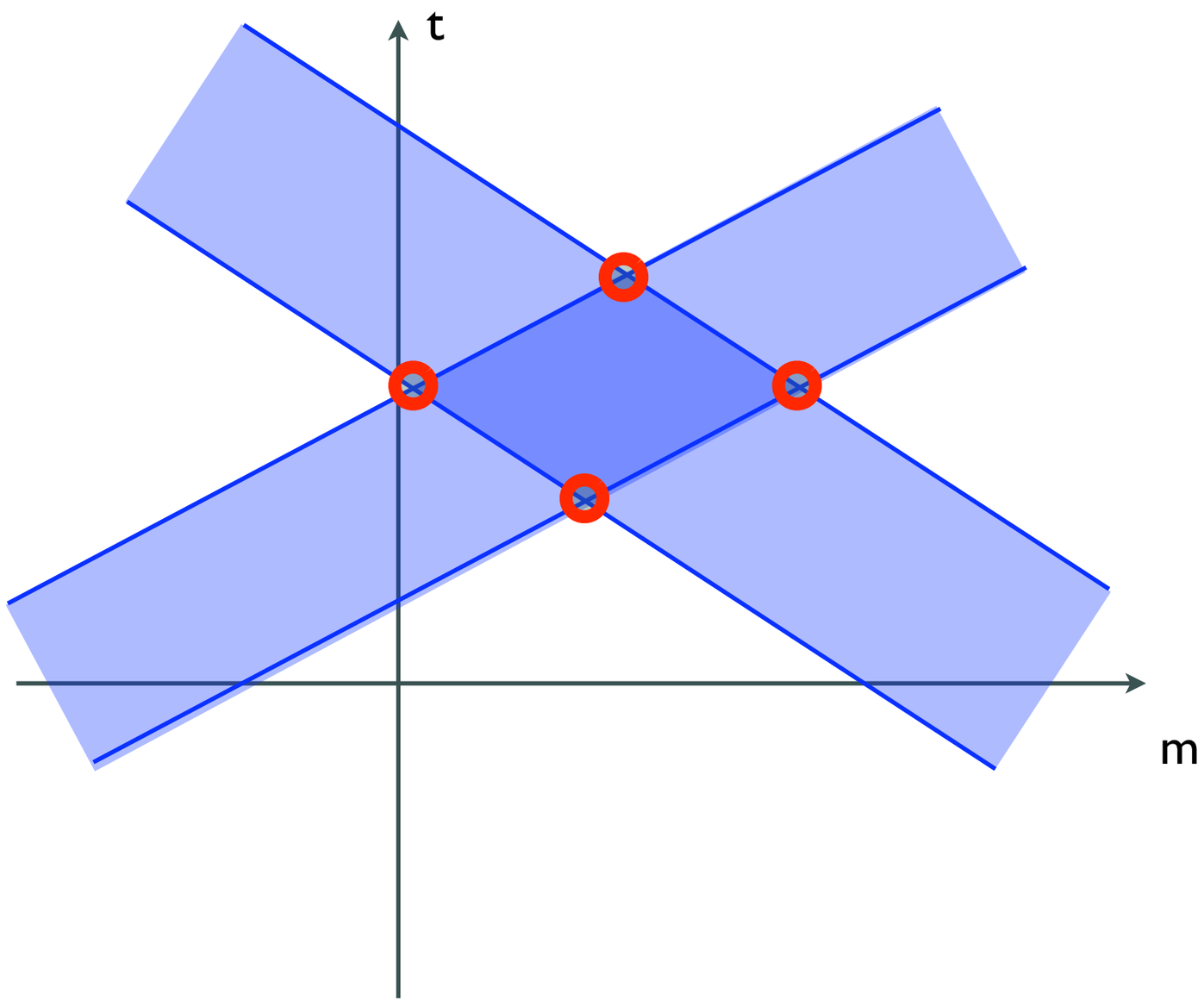
Hough Transformation: Variable binsize

- Use lines limiting the bands
- Calculate intersections between all limiting lines
- Bin border = mid between 2 intersections in m (t)
- Fill bin if band traverses bin



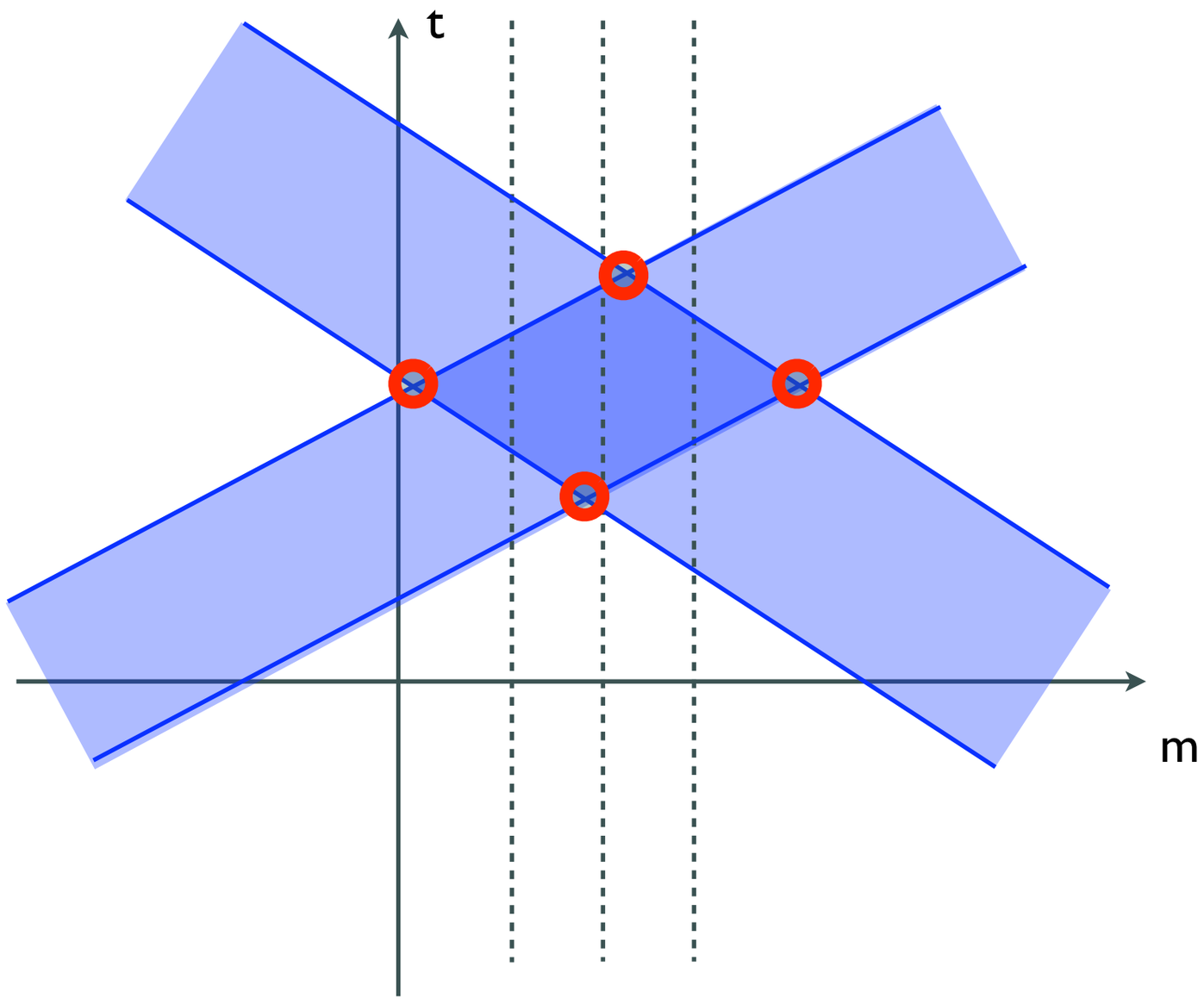
Hough Transformation: Variable binsize

- Use lines limiting the bands
- Calculate intersections between all limiting lines
- Bin border = mid between 2 intersections in m (t)
- Fill bin if band traverses bin



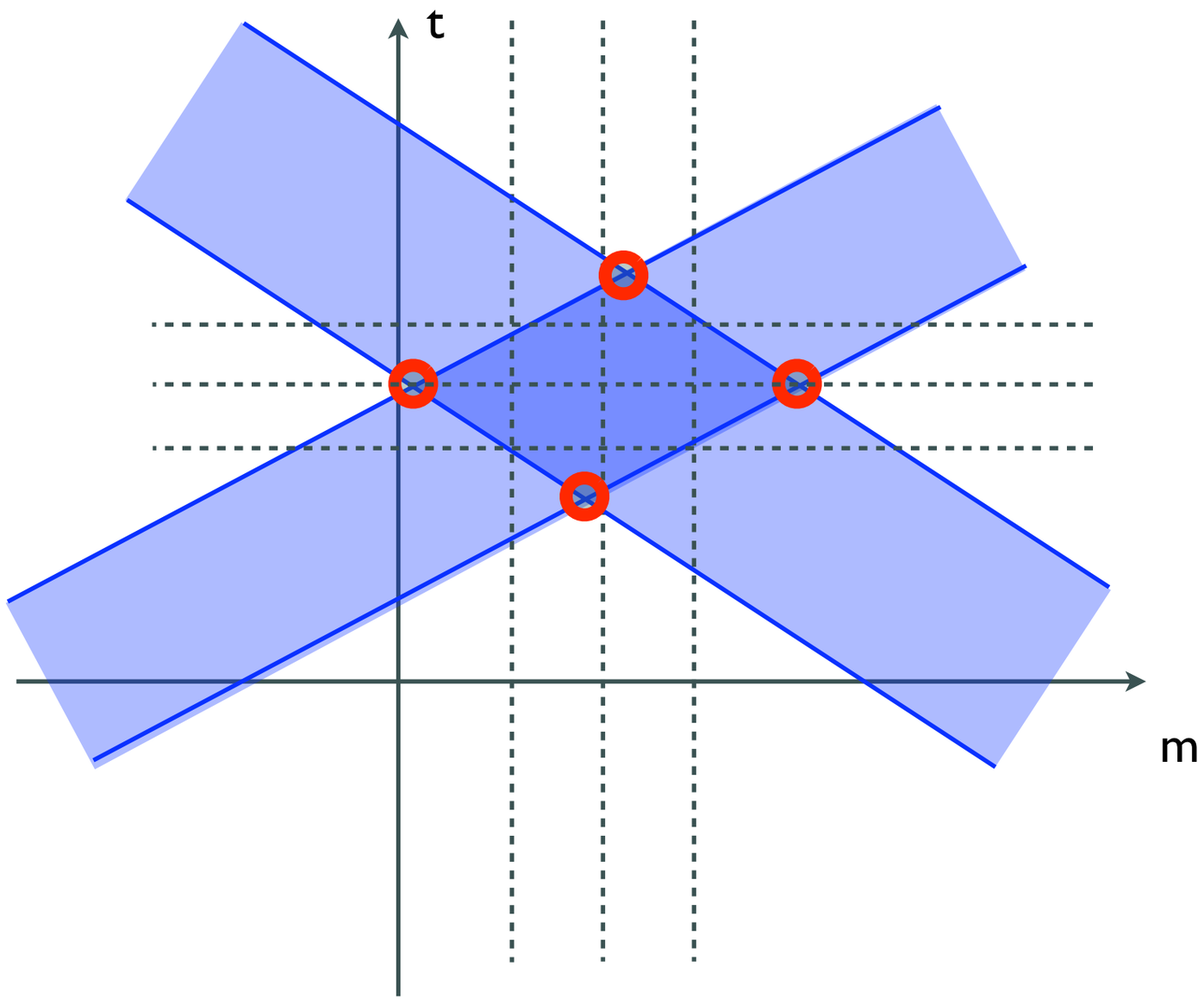
Hough Transformation: Variable binsize

- Use lines limiting the bands
- Calculate intersections between all limiting lines
- Bin border = mid between 2 intersections in m (t)
- Fill bin if band traverses bin

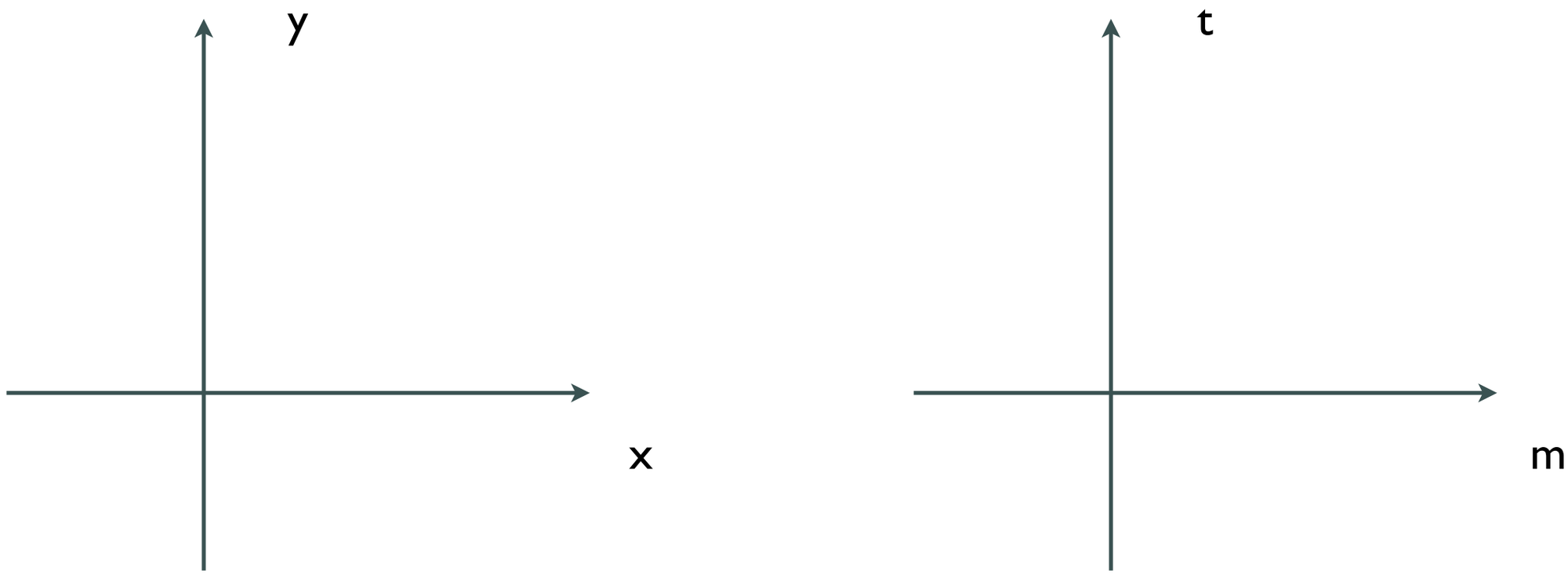


Hough Transformation: Variable binsize

- Use lines limiting the bands
- Calculate intersections between all limiting lines
- Bin border = mid between 2 intersections in m (t)
- Fill bin if band traverses bin



Hough Transformation: Principle

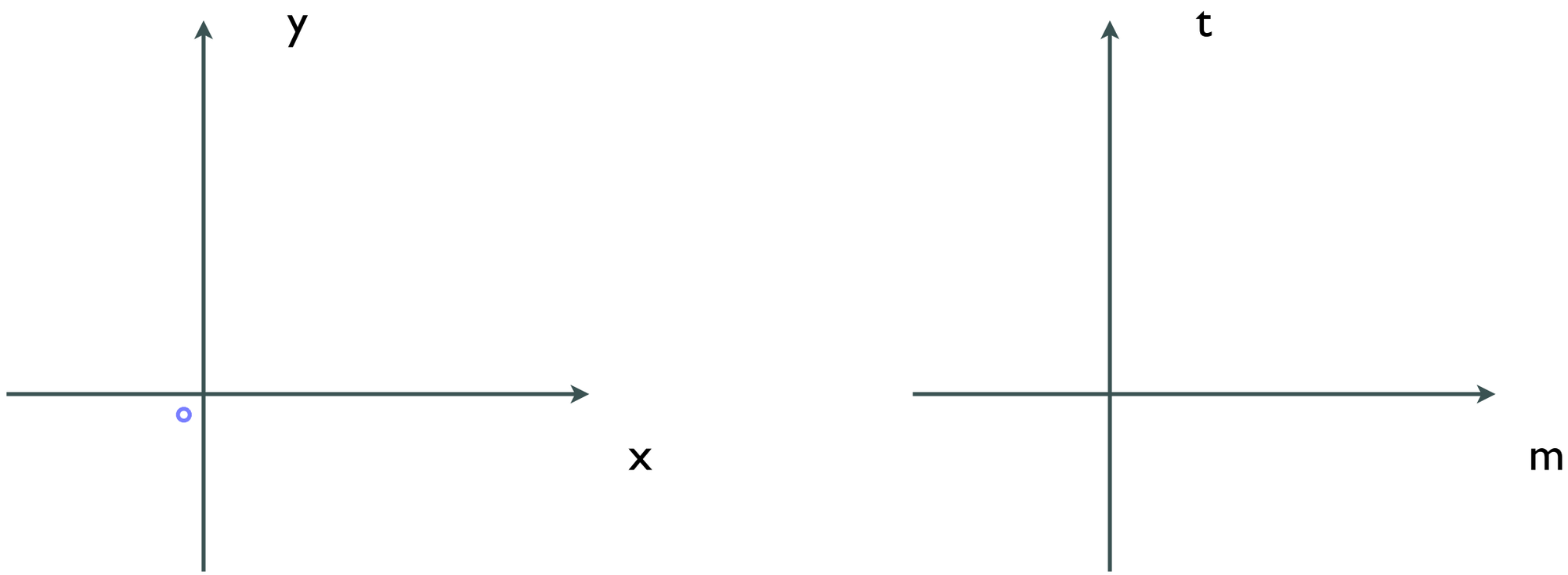


y = mx + t real space

t = -xm + y hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle

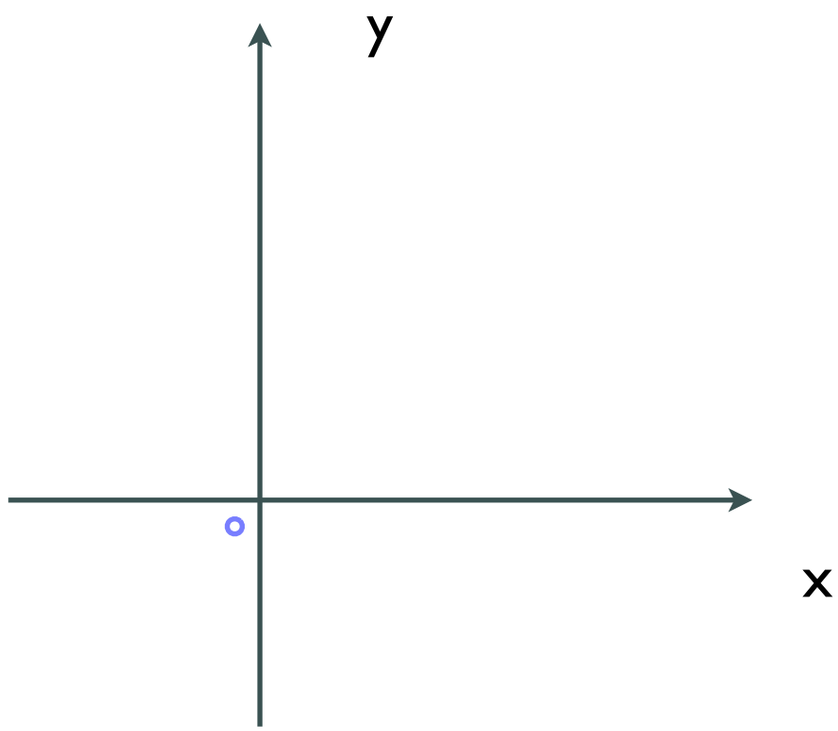


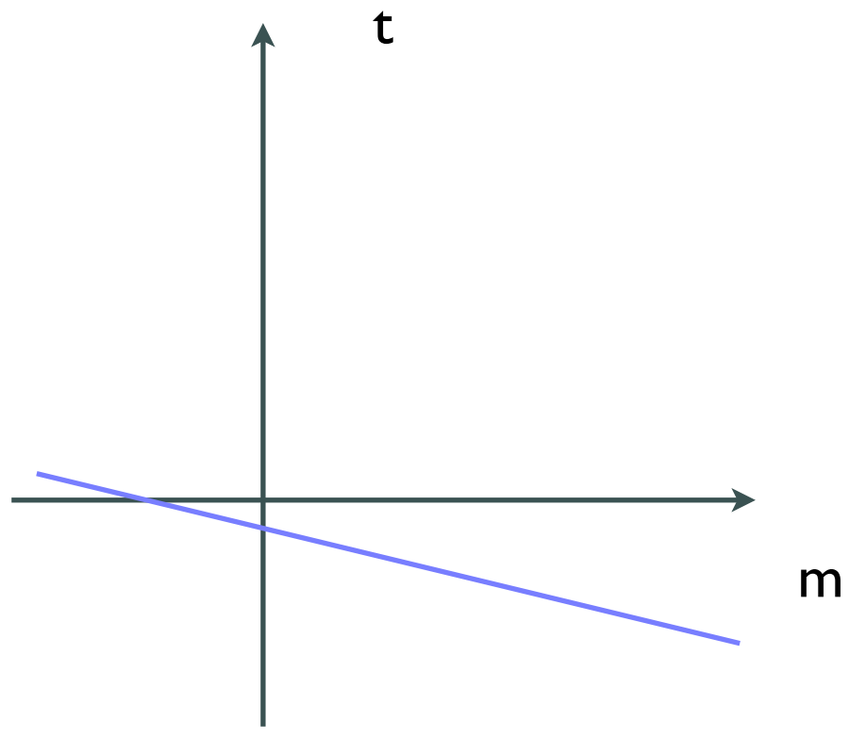
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



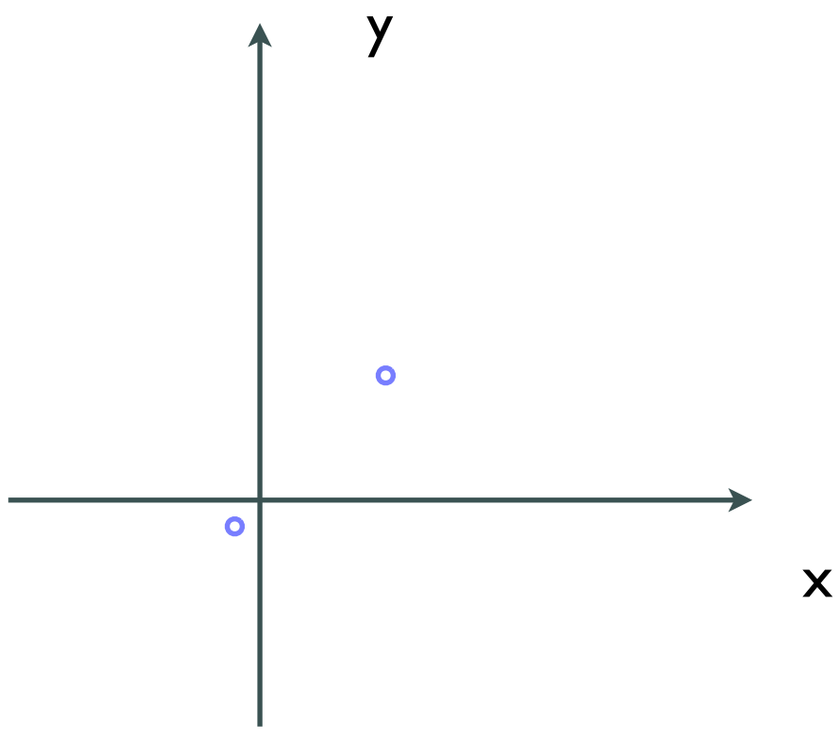


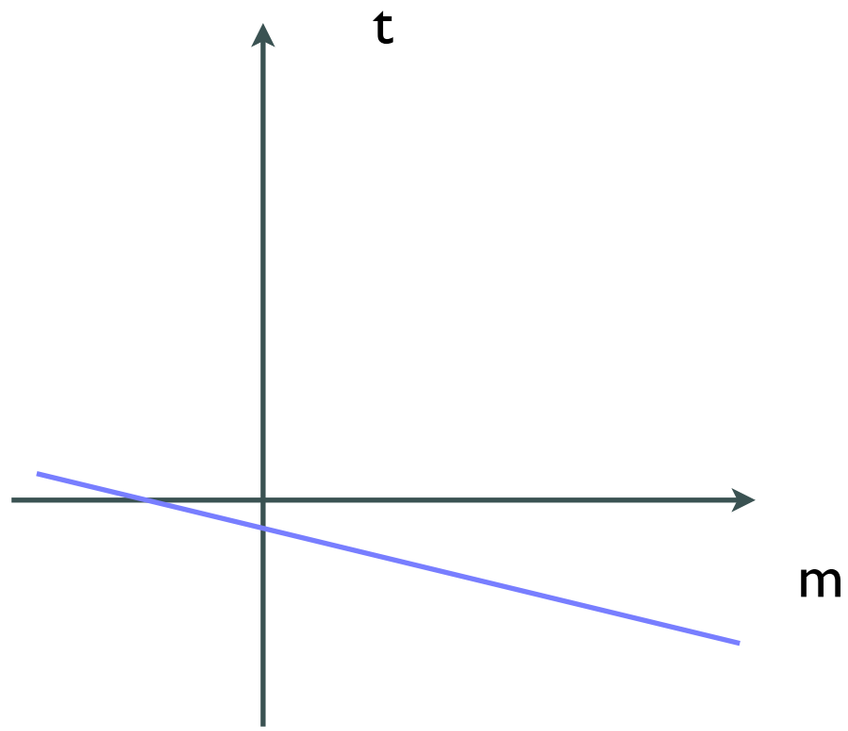
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



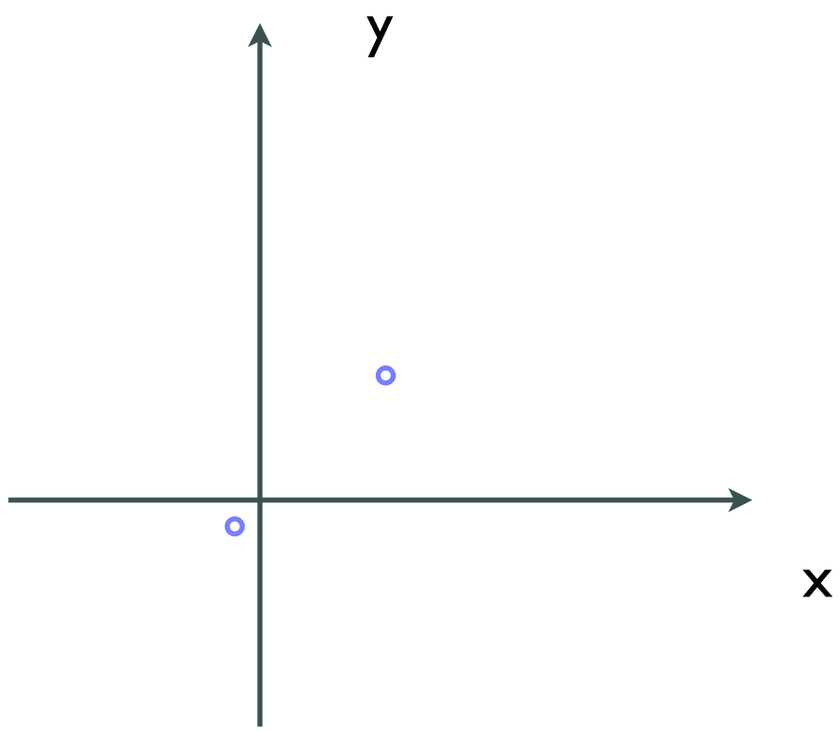


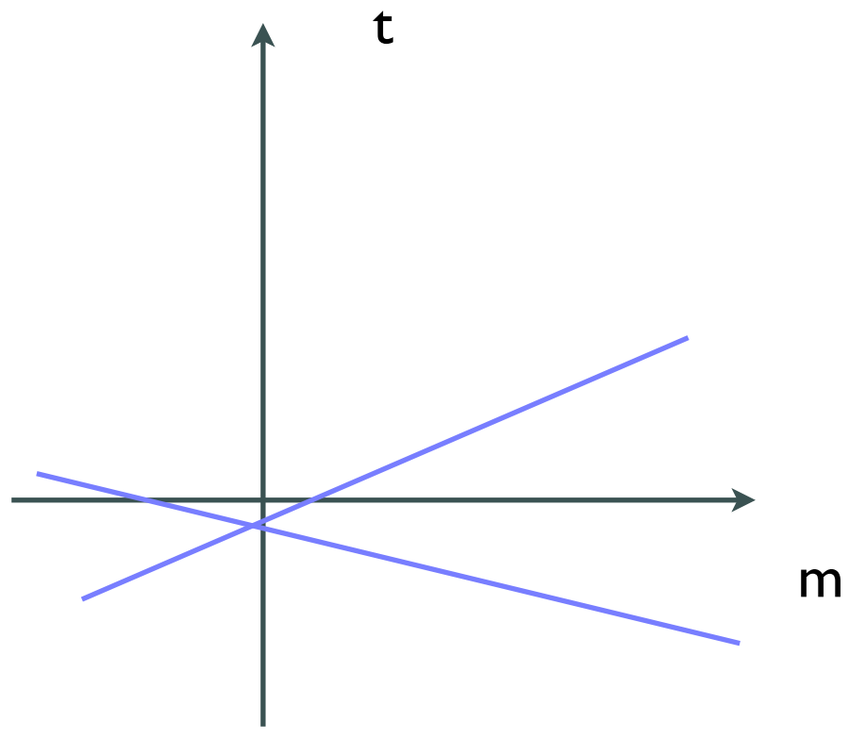
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



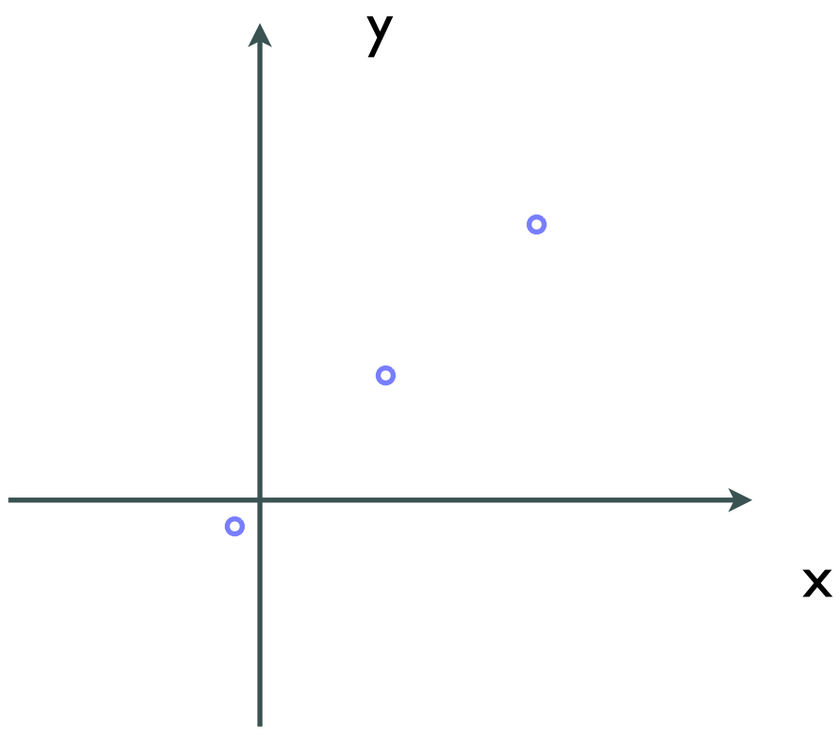


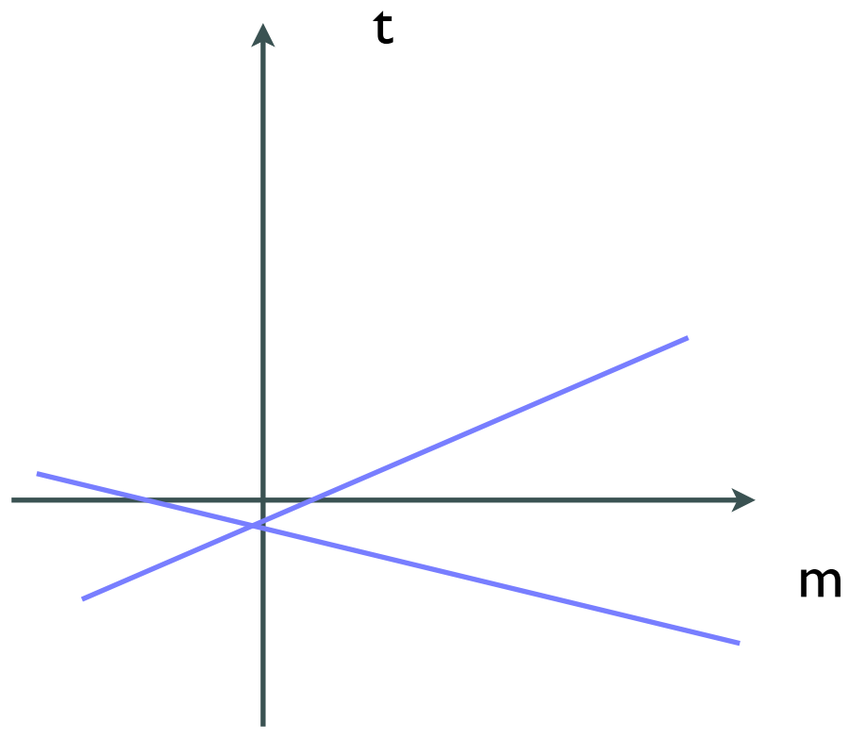
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



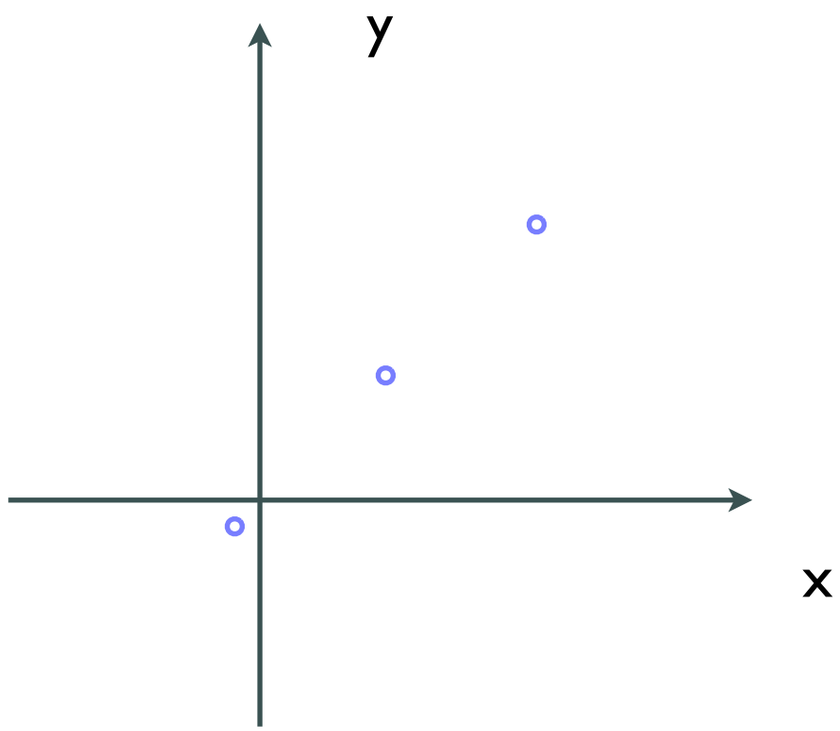


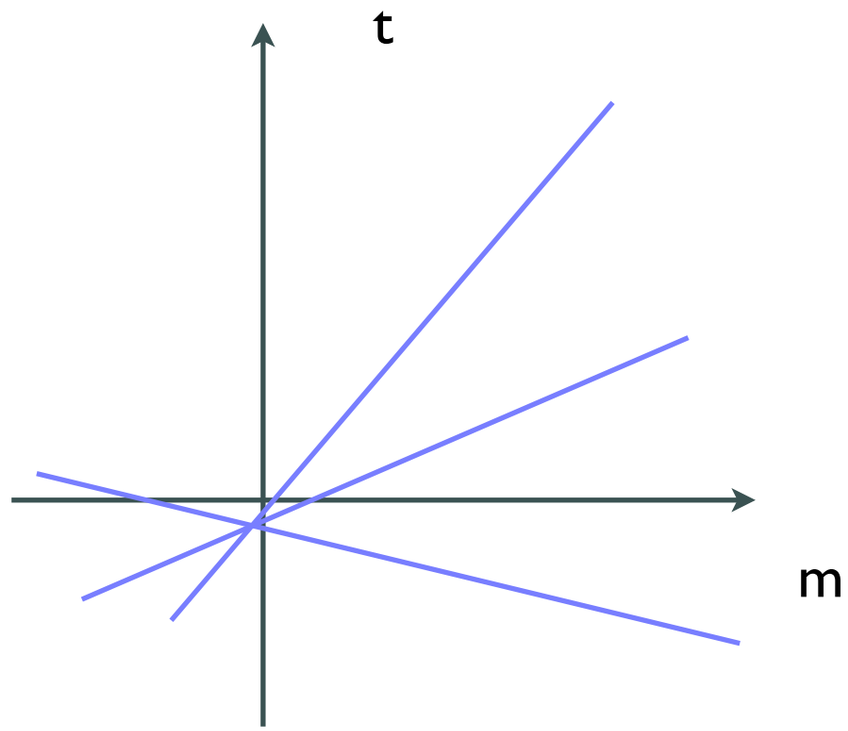
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



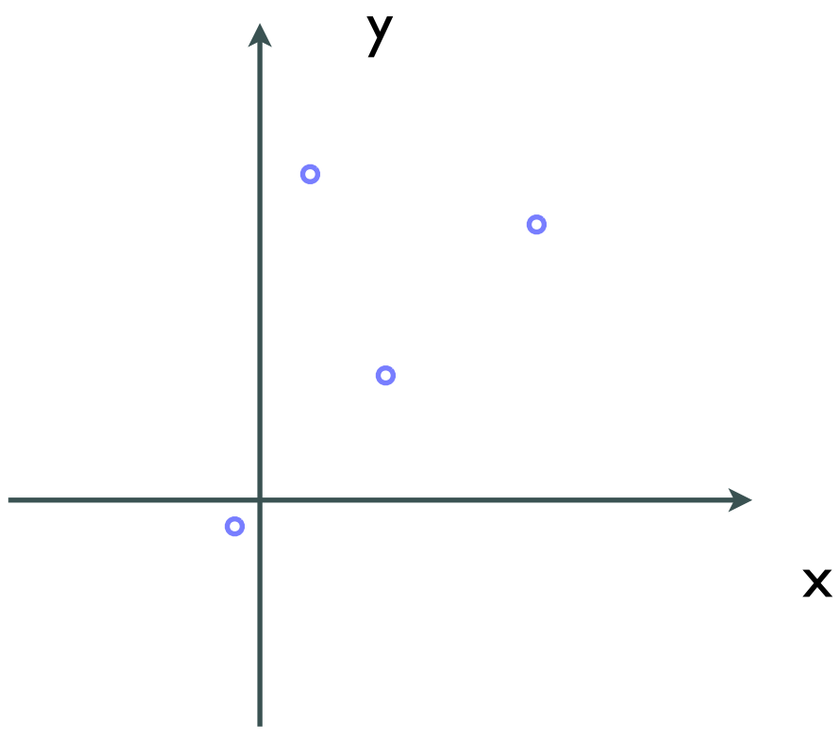


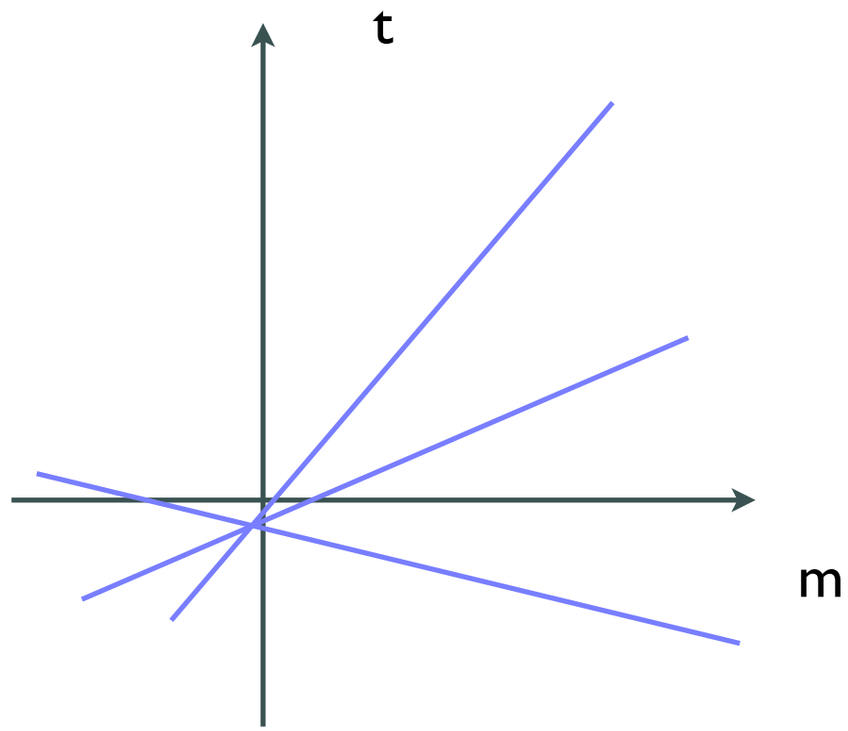
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



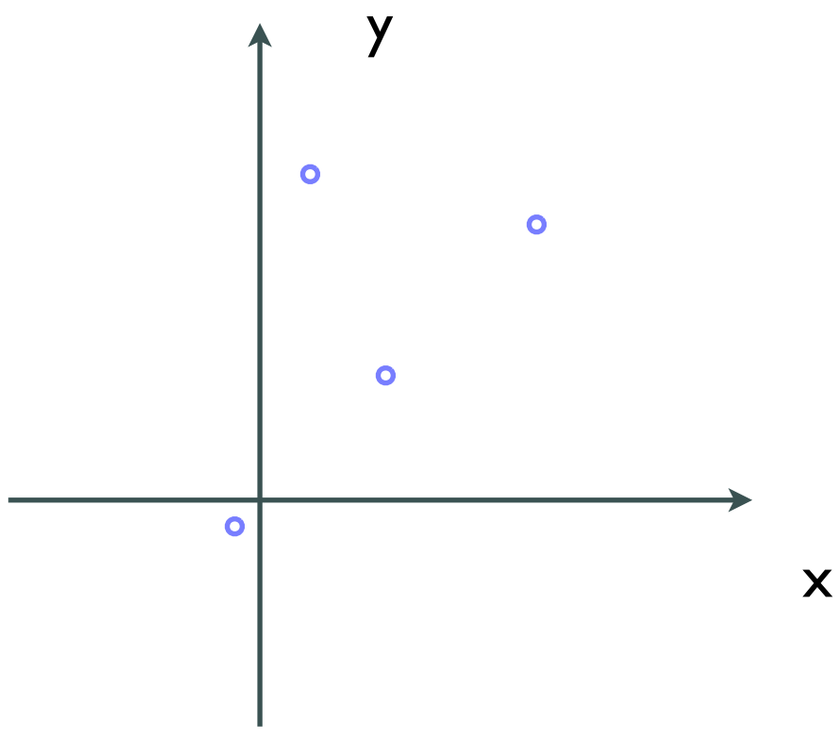


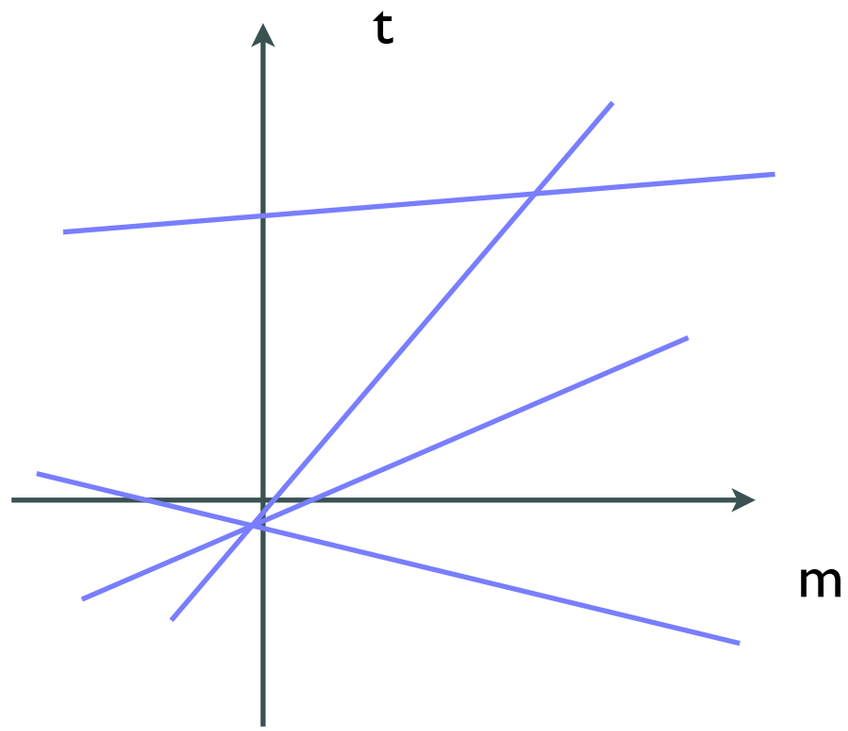
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



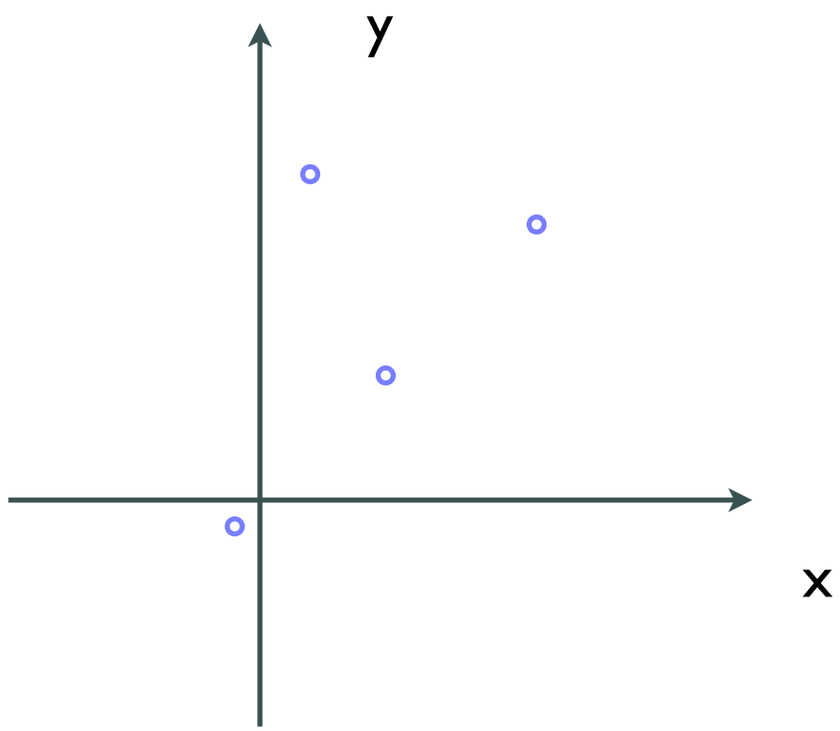


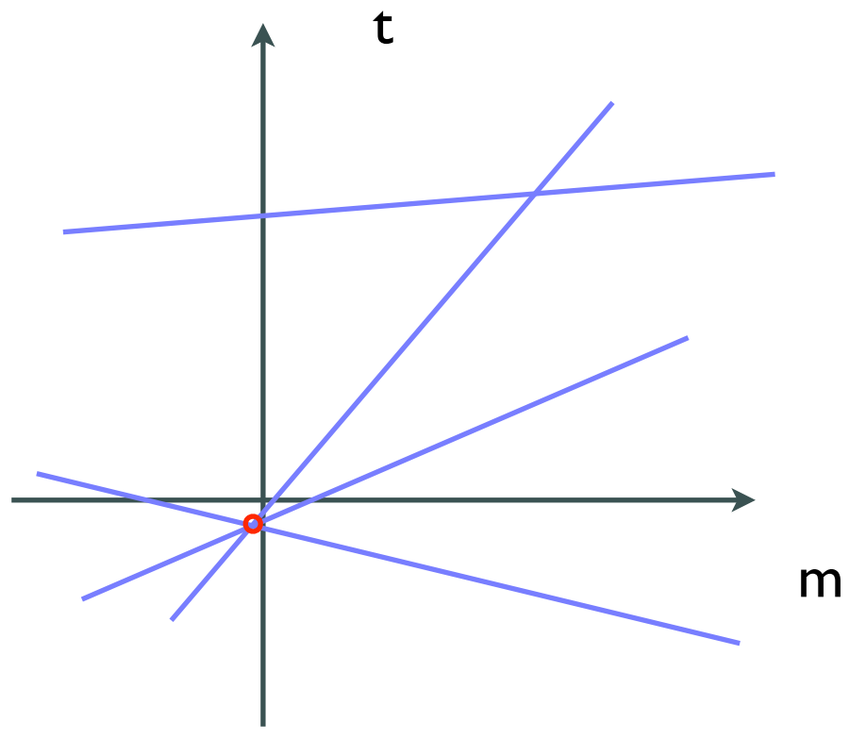
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle



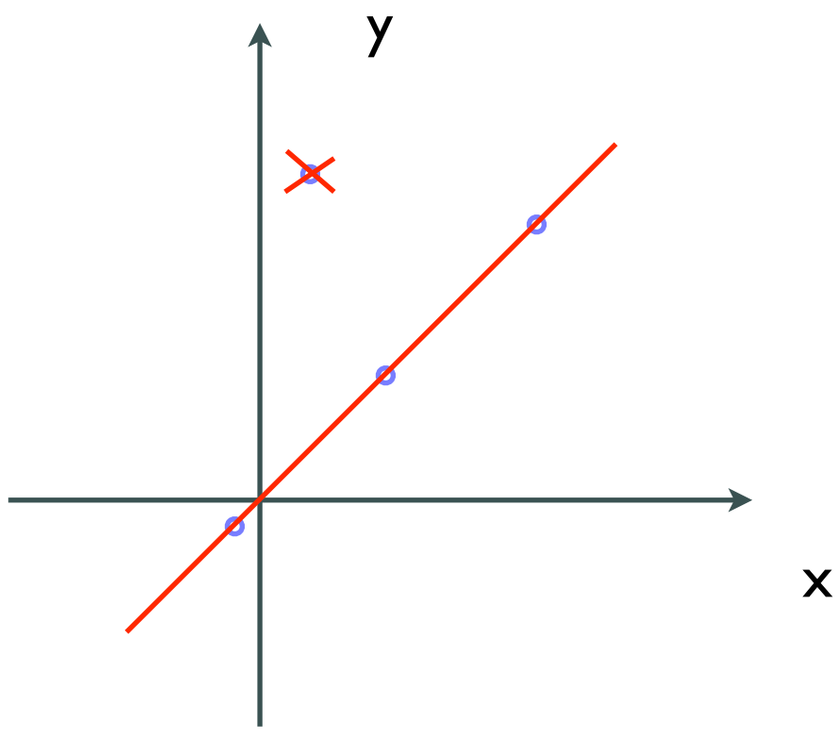


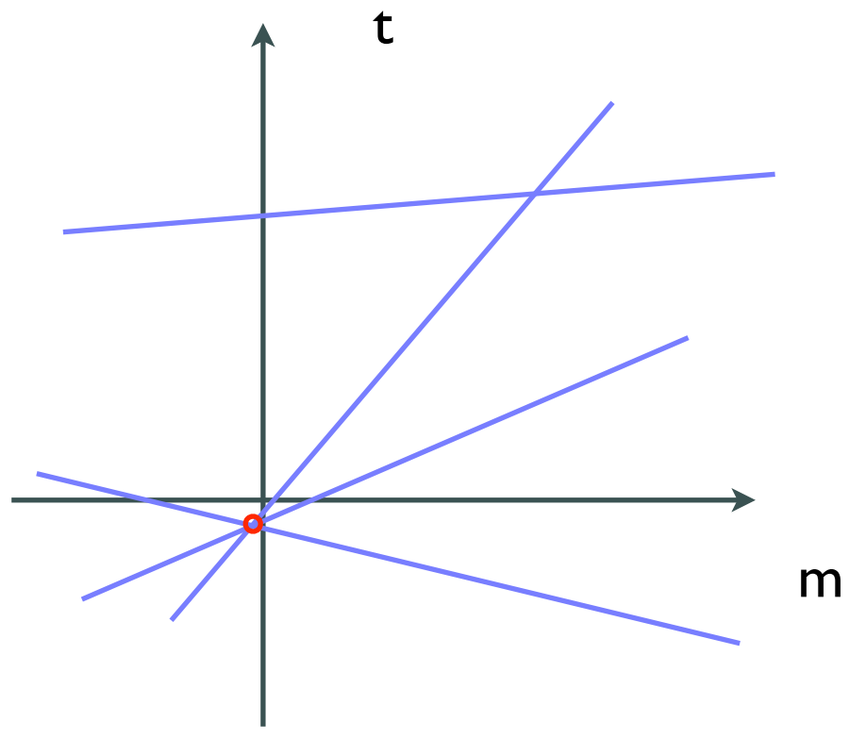
$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space

Hough Transformation: Principle





$y = mx + t$ real space

$t = -xm + y$ hough space

- Point in normal space = line in Hough Space (and vice versa)
- Get point with most intersections
  - Reject non corresponding points in real space