Track reconstruction for InGrid

Amir Noori Shirazi

Siegen University

ECFA- Linear Collider Workshop 2016
LCTPC: Pixel Meeting
01.06.2016

PHYSICS AT THE TERA SCALE
Helmholtz Alliance

UNIVERSITÄT SIEGEN

Outline:

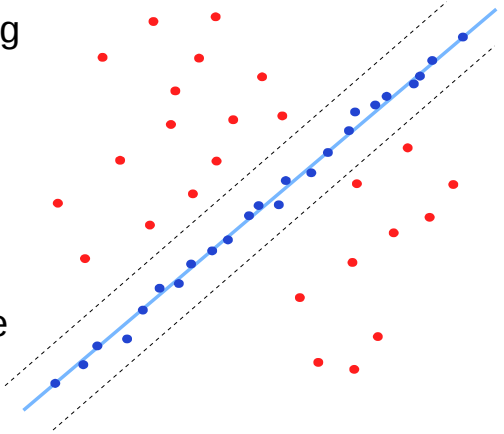
- **Goals**
- **RANSAC**
- **Super Pixel**
- **Linear regression**
- **Hough Transform**
- **Conclusion and outlook**

Goals:

- Try to find an algorithm for track finding for pixel TPC
- Use the unit of the chip as a segment of the track calling **Tracklet**
- Find the tracklets and then using Kalman filter or **G**eneral **B**roken **L**ines for connecting candidate tracklets

RANSAC: Introduction

- **Random Sample Consensus (RANSAC)**
  - Iterative method for estimating parameters of a model from a set of data containing outliers e.g. noise
  - Model given by physics case e.g. straight line
  - RANSAC tests this model by randomly choosing points from data set
  - Number **n** of randomly chosen points depends on the model, **n** = 2 for straight line
  - Repeat procedure **N** times and keep all solutions fulfilling threshold
  - **Inliers:** all points with distance to line fit smaller than tolerance
  - **Tolerance:** width of the distribution of the hits from the model
  - **Threshold:** fraction of inliers greater than given value **W**
  - **W:** number of inliers / number of points in data set
  - **Consensus:** the set of inliers for the fitting model
  - Finding the best fit based on the χ^2



By Msm - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2071406

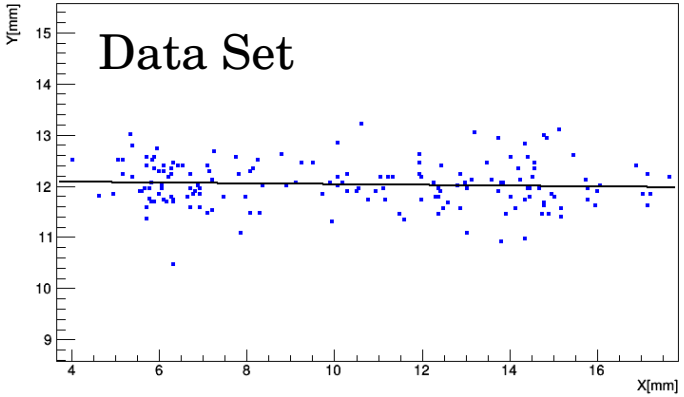
$$N = \frac{\log(1-P)}{\log(1-w^n)}$$

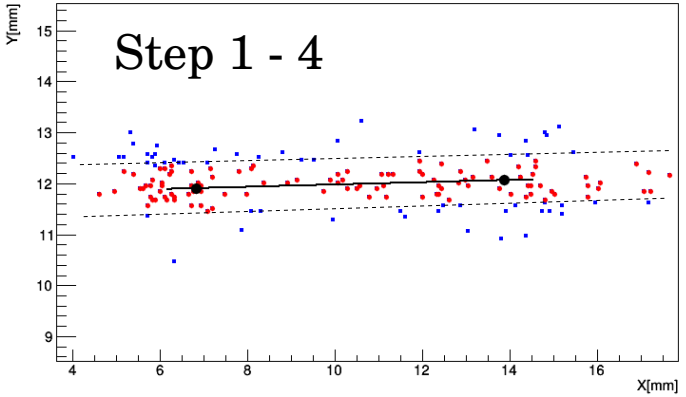
P: probability of finding solution with N iterations (99%)

W	N
0.1	458
0.2	112
0.3	48
0.4	26
0.5	16
0.6	10
0.7	6
0.8	4
0.9	2
1	0

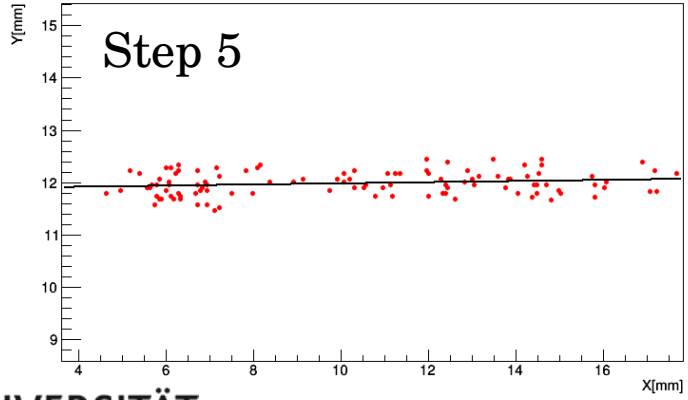
RANSAC: MarlinTPC

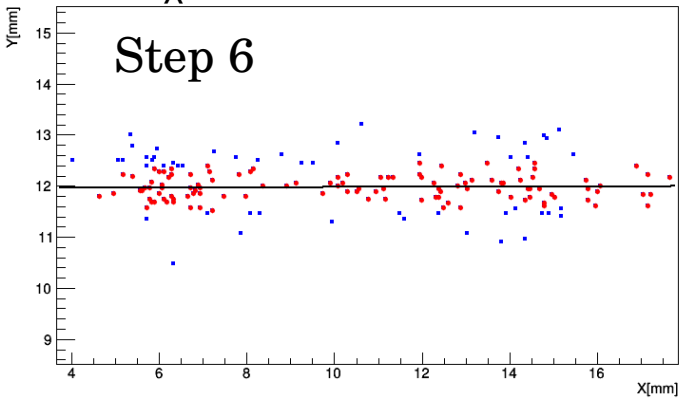
1) Choose two random hits
2) Define straight line based on these two hits
3) Calculate the distance of all other hits to the line





4) Collect those points with the distance less than the tolerance
5) If there are enough points in collection then a solution is found. Fit the line using all points in collection.
6) Recalculate inliers and outliers points based on this fit and then refit again based on new inlier points.
7) Repeat step 1 to 6 N times.
8) If more than one solution is found take solution with smallest χ^2.



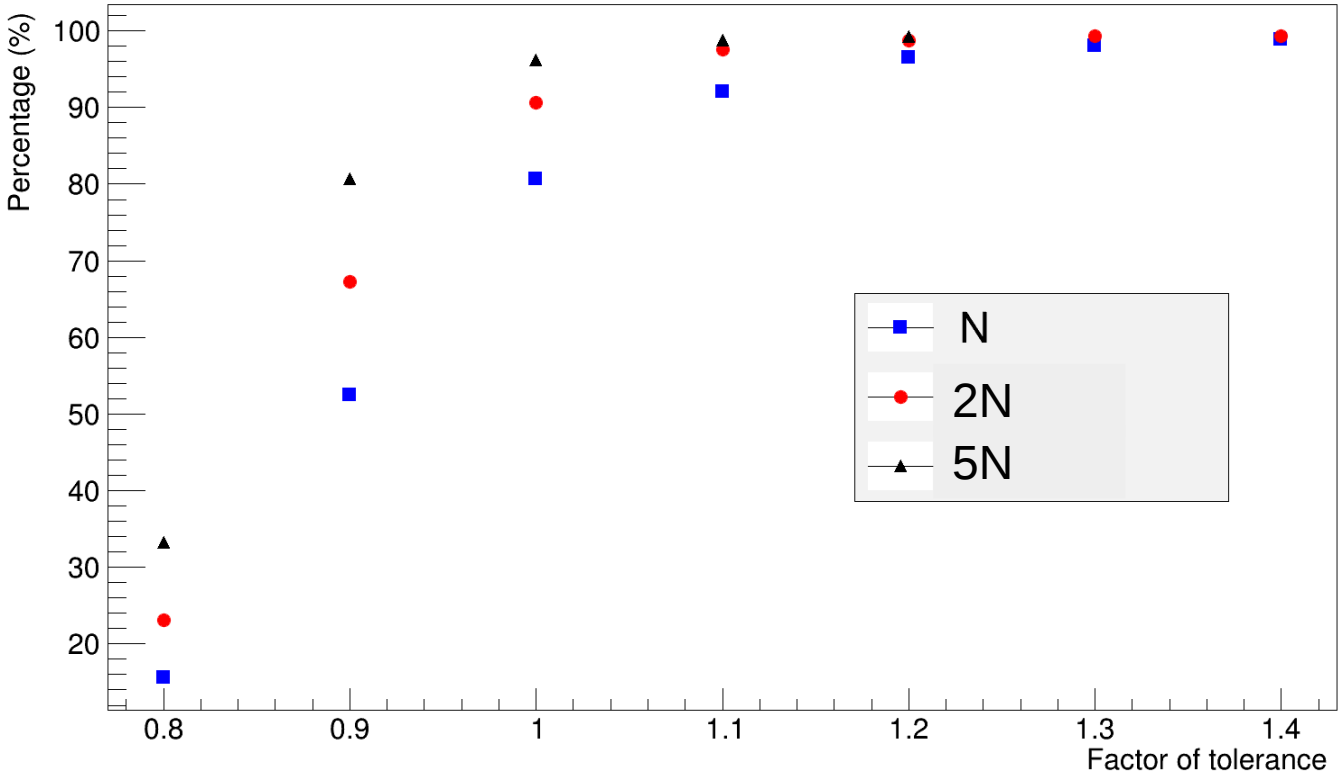


RANSAC: MarlinTPC

- **Optimization:**
  - Simulation of 1000 Events:
    - One InGrid Chip
    - 20 mm distance of the track from the readout plane
    - Parallel to the x axis from Y = 12 mm
    - The straight line model $Y = P0 + P1\,X$
    - W= 0.6 => N=10
  - Use width of the track as the tolerance (In this case ~0.44 mm)
  - Width of the track is calculated based on the distance of the track from the readout plane and diffusion coefficient.
  - Analysis done for different scaling factors of width of the track (0.8,0.9,...,1.4)
  - Results: Distribution of P0 and P1 for all 1000 events
  - For few events solution is found using scaling factor of 0.8 and for around 98% of events solution is found for scaling factor 1.4 with standard N iteration.
  - In order to find the optimal tolerance, the previous step has been repeated for two larger number of Iterations.

RANSAC: Number of Iteration

Percentage of events with at least one solution for different number of iterations



$$N = \frac{\log(1-P)}{\log(1-w^n)}$$

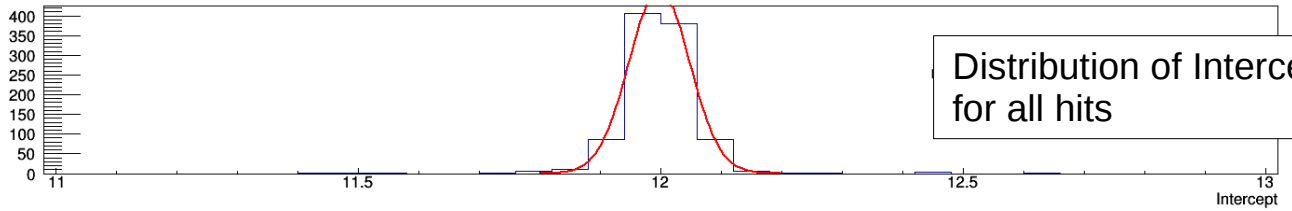
Expected percentage is 99%

1.0	# candidate (%)
N = 10	80.7
2N	90.6
5N	96.2

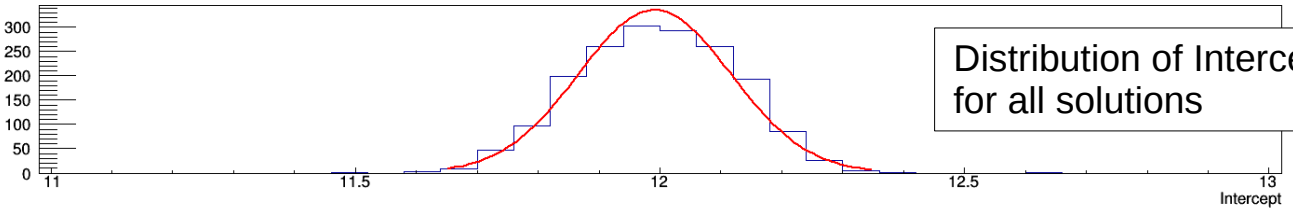
- Keep the tolerance small in order to have better two track resolution
- In our case N iterations are not sufficient to reach optimal percentage.

RANSAC: Result for scaling factor of 1.0

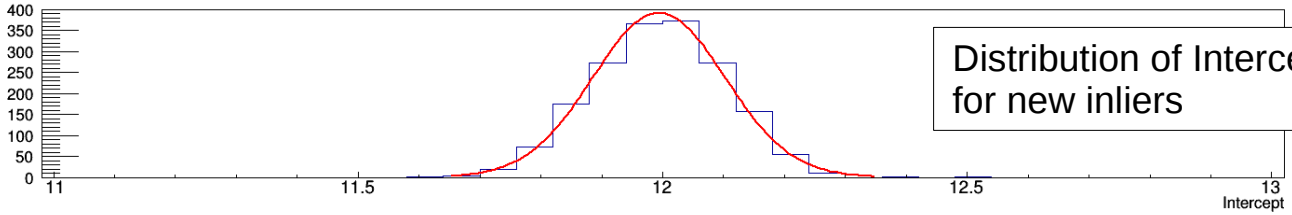
Fit on all hits



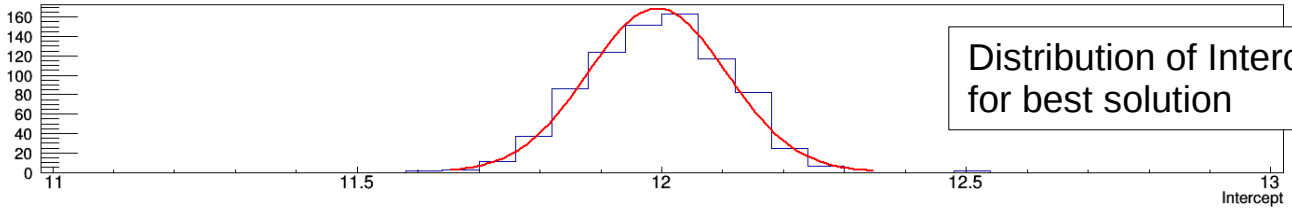
Distribution of Intercept (p0) for all hits



Distribution of Intercept (p0) for all solutions



Distribution of Intercept (p0) for new inliers



Distribution of Intercept (p0) for best solution

	Nominal value [mm]	Mean value all solutions [mm]	Sigma all solutions [mm]	Number of solutions	Mean value best solution [mm]	Sigma best solution [mm]	Number of Best solution
p0	12.0	11.99 +/- 0.00	0.125 +/- 0.002	1781	11.99 +/- 0.00	0.114 +/- 0.003	807
p1	0.0	0.000054 +/- 0.007	0.027 +/- 0.0005	1781	0.00033 +/- 0.0010	0.028 +/- 0.0008	807

Super Pixel: Introduction

SuperPixel:

- Offered by **Claus Kleinwort**
- Each Super pixel includes several pixels.
  - $a=2^n$ Number of pixels in each row and column
  - $w=256/a$ w^2 Number of super pixel for each chip

- Collect all hits inside of the each super pixels → calculate center of gravity→ **SuperHit**
- CellID0 → SuperPixel number
- CellID1 → Chip number
- Covarinace Matrix →Calculated based on the super pixel
- Edep → number of hits in each SuperHit

Tracklet finding:

- Linear regression: $Y = X\beta + \epsilon \Rightarrow \beta = (X^T D^{-1} X)^{-1} X^T D^{-1} Y$

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \begin{matrix} \rightarrow \text{Intersect} \\ \rightarrow \text{Slope} \end{matrix}$$

- Hough Transform: $d_0 = y\cos(\varphi) - x\sin(\varphi)$

Covariance Matrix for a Super pixel:

Covariance matrix for all hits in each super pixel for x,y,z

$$V_{ij} \equiv cov(X_i, X_j) \equiv \langle (X_i - \bar{X}_i)(X_j - \bar{X}_j) \rangle$$

Spacial Transverse resolution for super hits

$$\sigma_D^2 = \left(a . \frac{LPix}{\sqrt{12}}\right)^2 + \frac{ZD_T^2}{N}$$

Spacial Longitudinal resolution for super hits

$$\sigma_L^2 = 16 + \frac{(T_c v_d)^2}{12} + \frac{ZD_L^2}{N}$$

Covariance matrix for super hit

$$G = (V_{ij} + \sigma_D^2 + \sigma_L^2)$$

Lpix : length of a pixel (0.055 mm)

a: number of pixel in each row of a super pixel

N: number of hits

$\tau_c = 1/f_c$, $f_c = 40$ MHz

G: 3x3 matrix

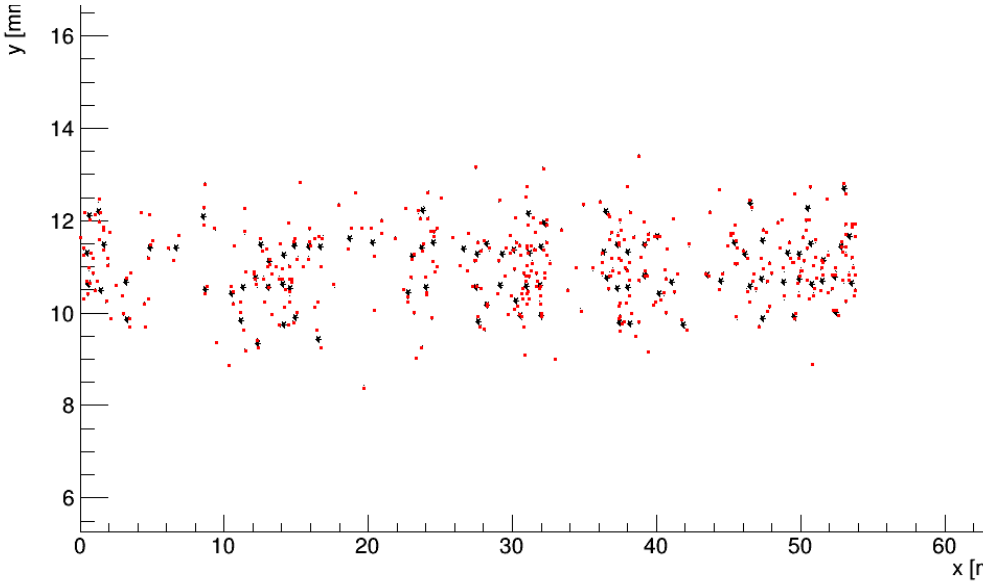
Super Pixel: Simulation

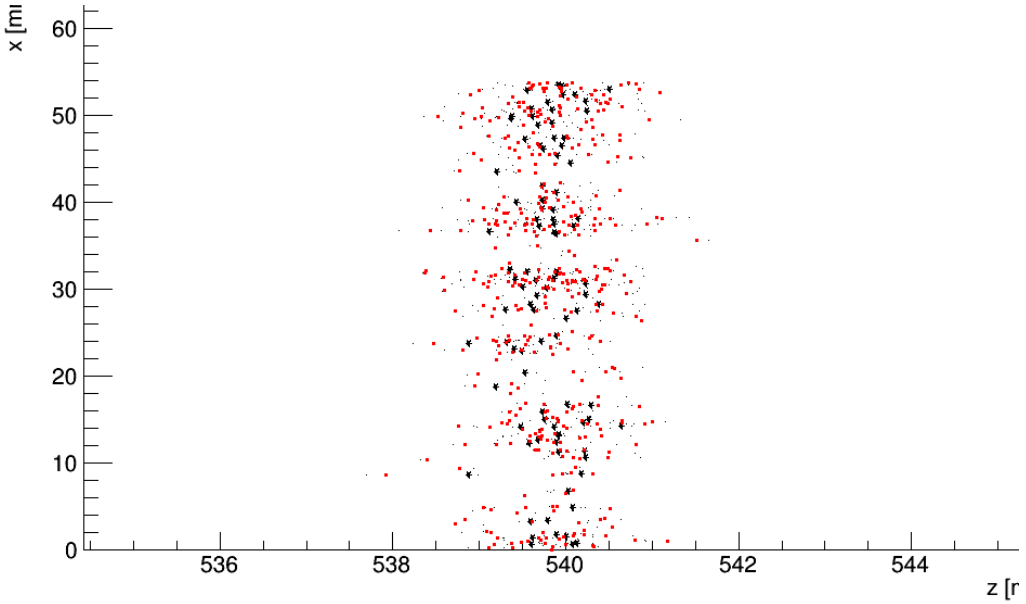
Simulation 1000 Events:

- Y0 = 12 mm , parallel to X-axis
- Z = 60 mm from the readout plane
- Weighted Least Square (XY- Plane): $G = V_{22} + \sigma_D^2$
- Weighted Least Square (ZX- Plane): $G = V_{33} + \sigma_L^2$

$\Rightarrow D_{ij} = G_i \delta_{ij}$

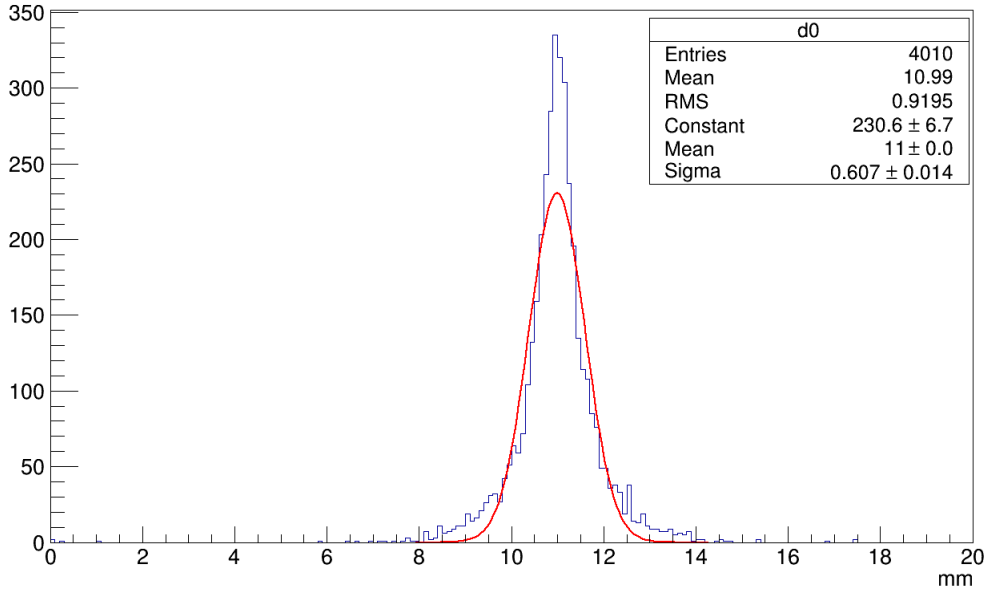
- 16 x 16 pixels in each super pixel



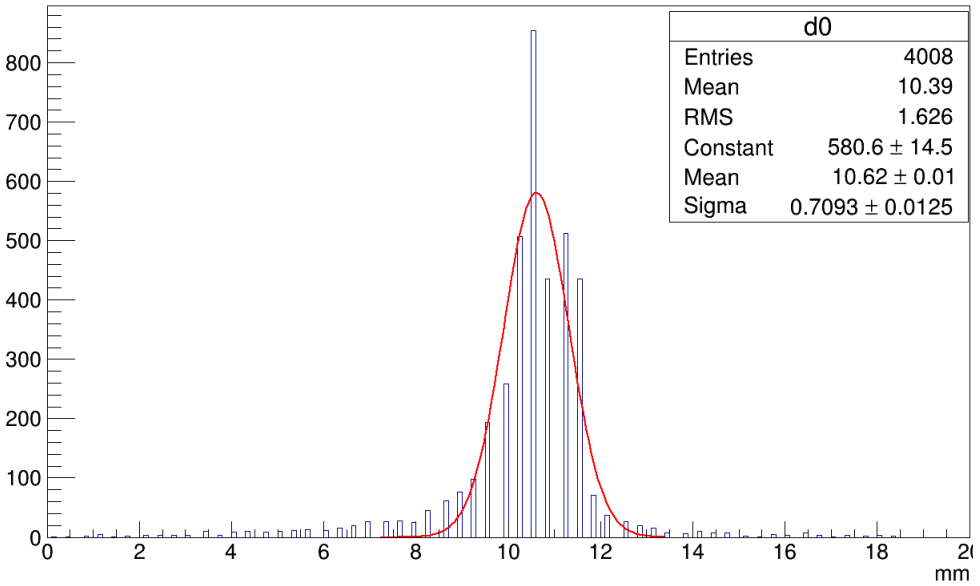


Super Pixel: Linear Regression & Hough Transform

d0 from linear regression

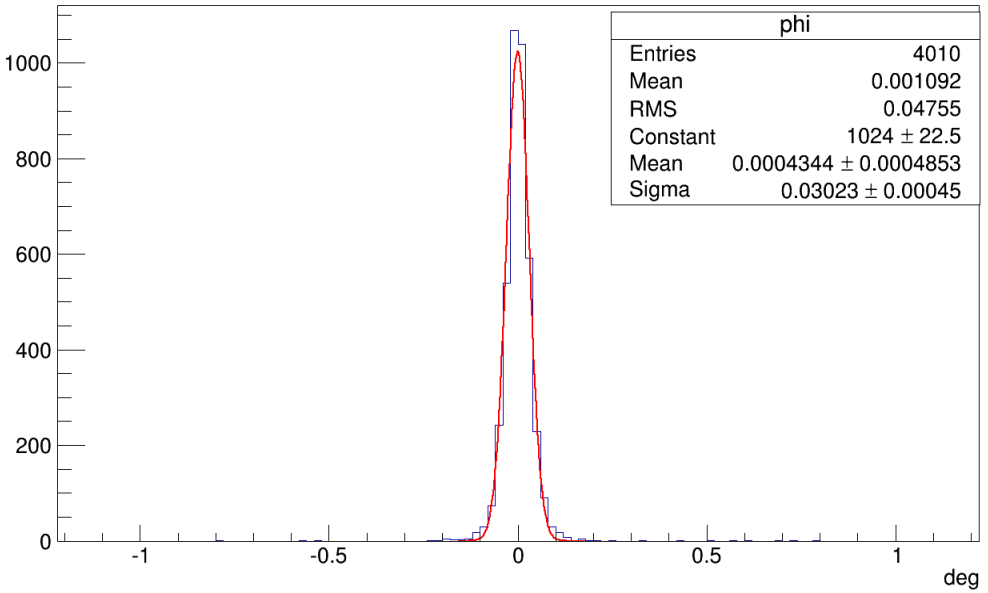


d0 from Hough Transform

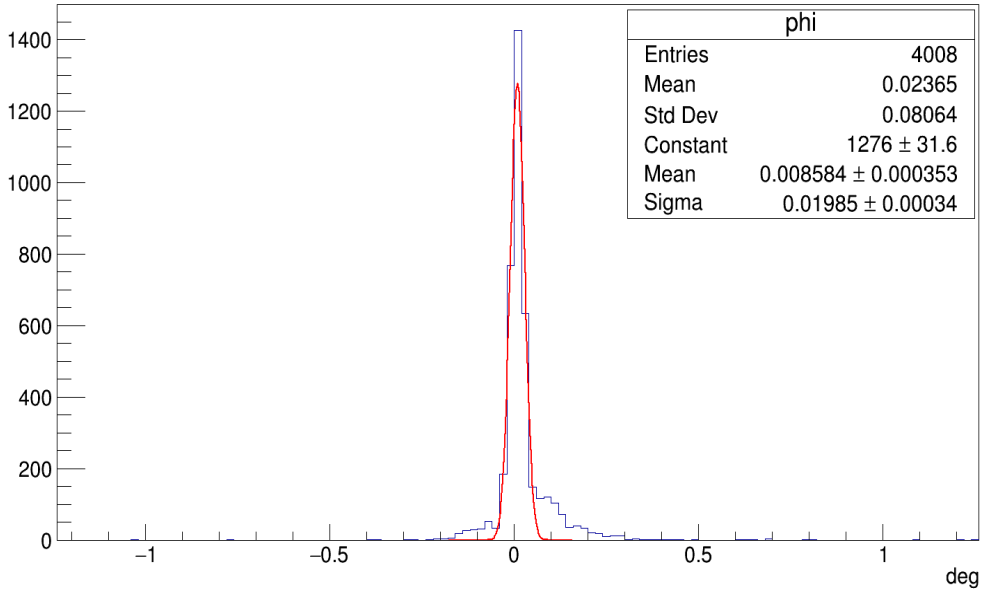


Super Pixel: Linear Regression & Hough Transform

φ from linear regression



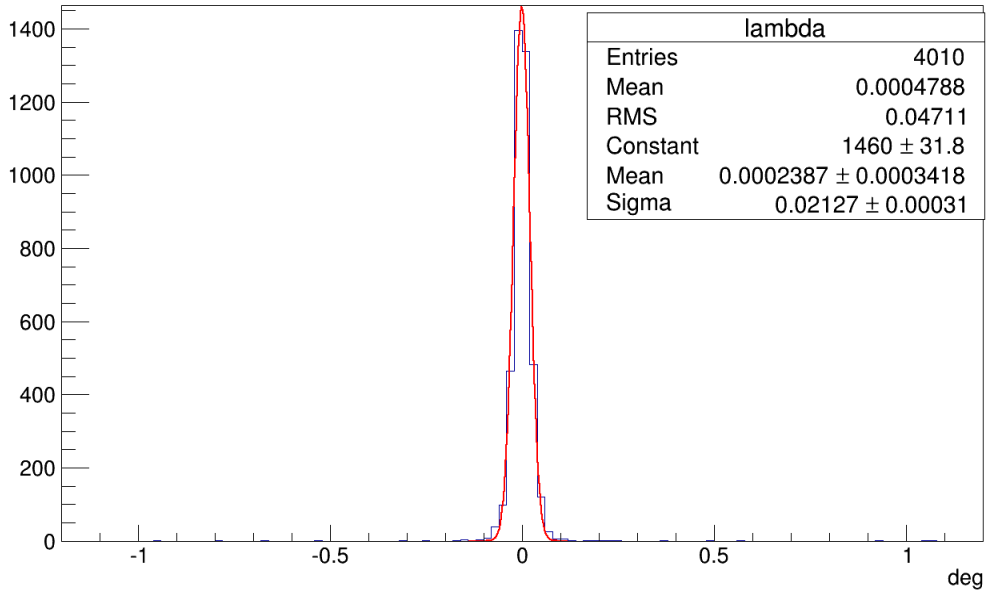
φ from Hough Transform



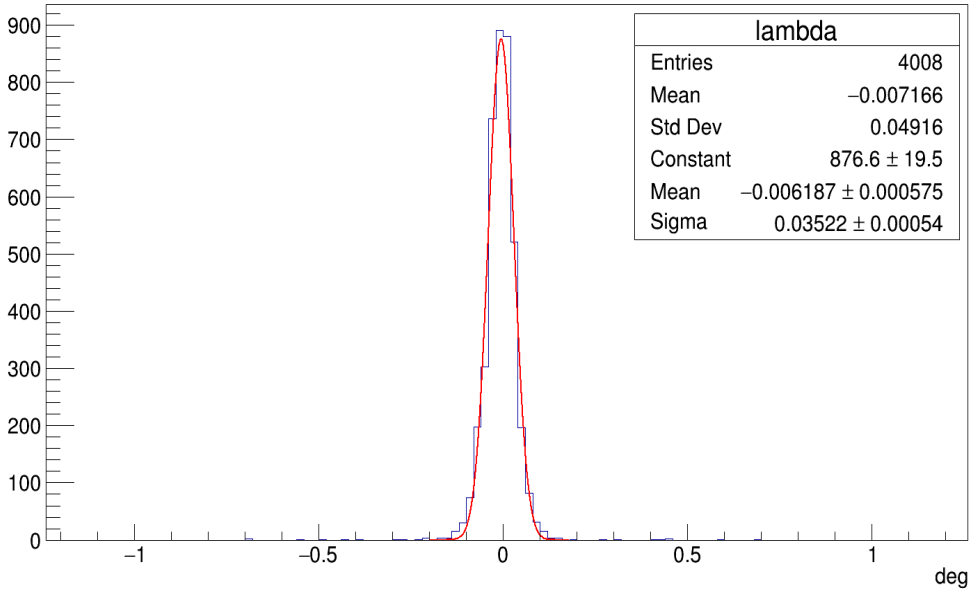
UNIVERSITÄT SIEGEN

Super Pixel: Linear Regression & Hough Transform

λ from linear regression



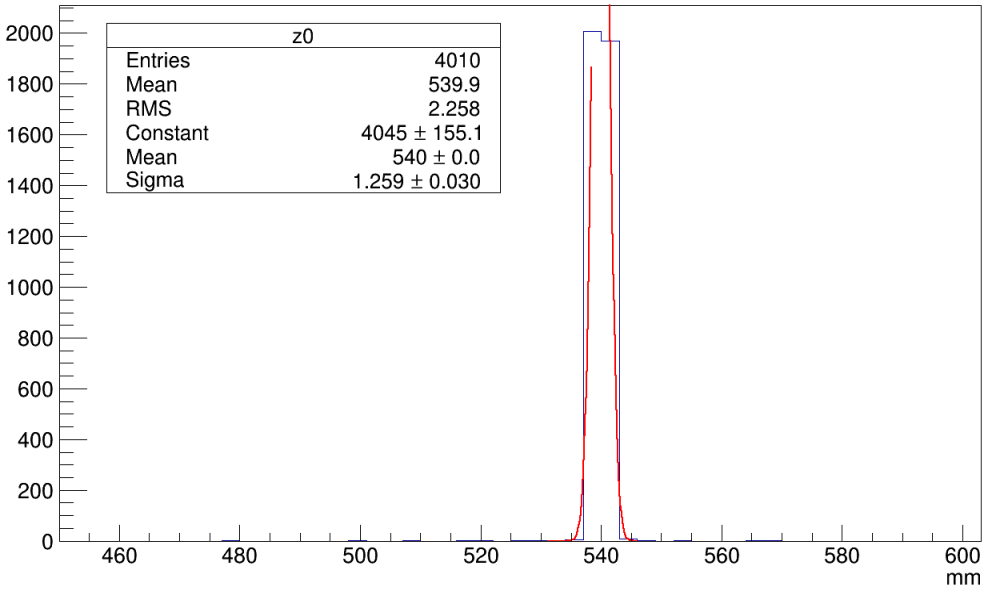
λ from Hough Transform



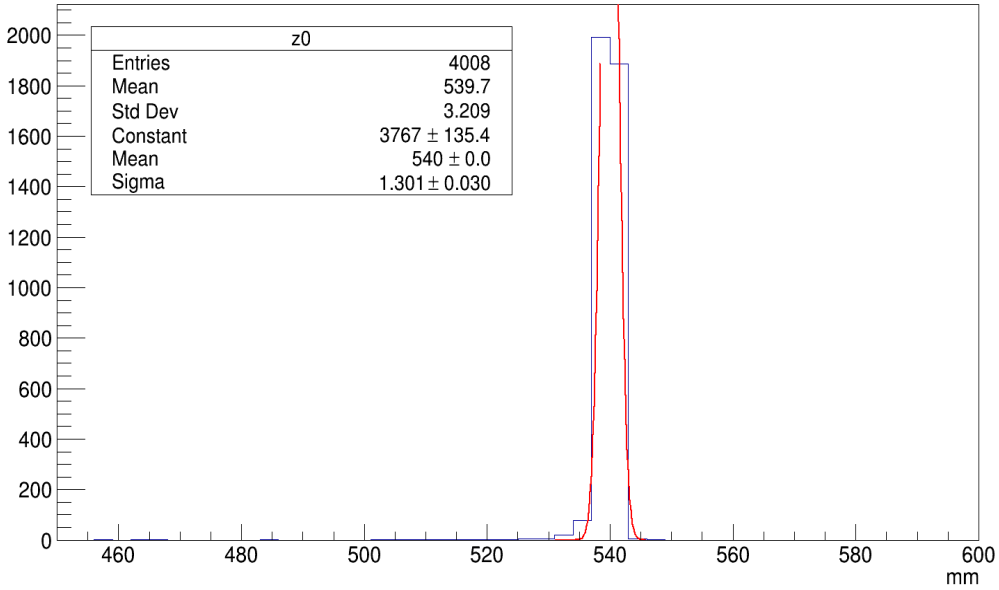
UNIVERSITÄT SIEGEN

Super Pixel: Linear Regression & Hough Transform

z0 from linear regression



z0 from Hough Transform



UNIVERSITÄT SIEGEN

Summary & Outlook

- **Conclusion:**

- RANSAC for many chips is slow.

- Linear regression needs to know direction of the track. For φ> 45° should be interchanged.

- **Outlook:**

- Find the direction of a track with Hough Transform and then using linear regression

- Use RANSAC for super pixel → Less number of iteration

- Find inliers and outliers to remove noise

- Simulation of 2 tracks with various distances from each other

- Use real data from test beam using up to 160 chips