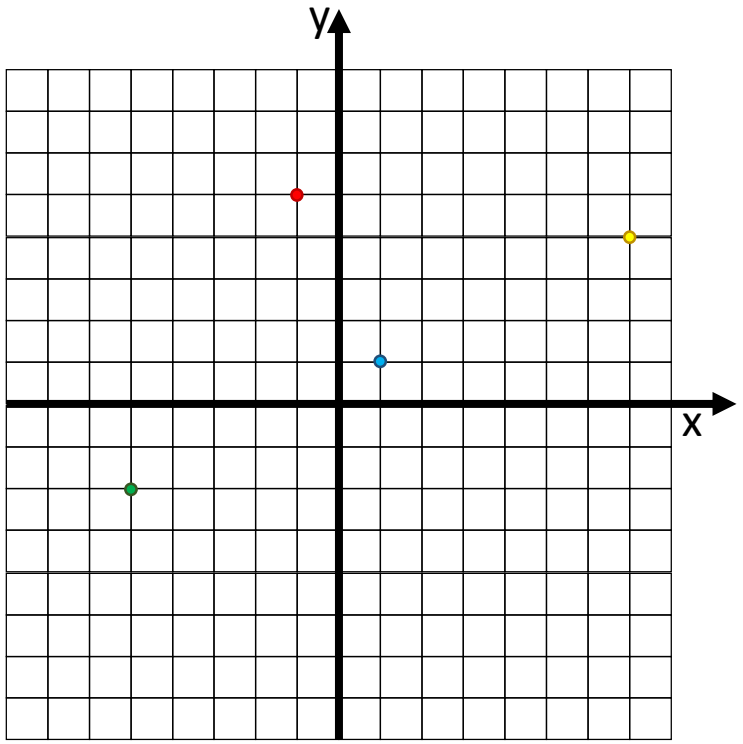
# Hough Transforms and GNNs

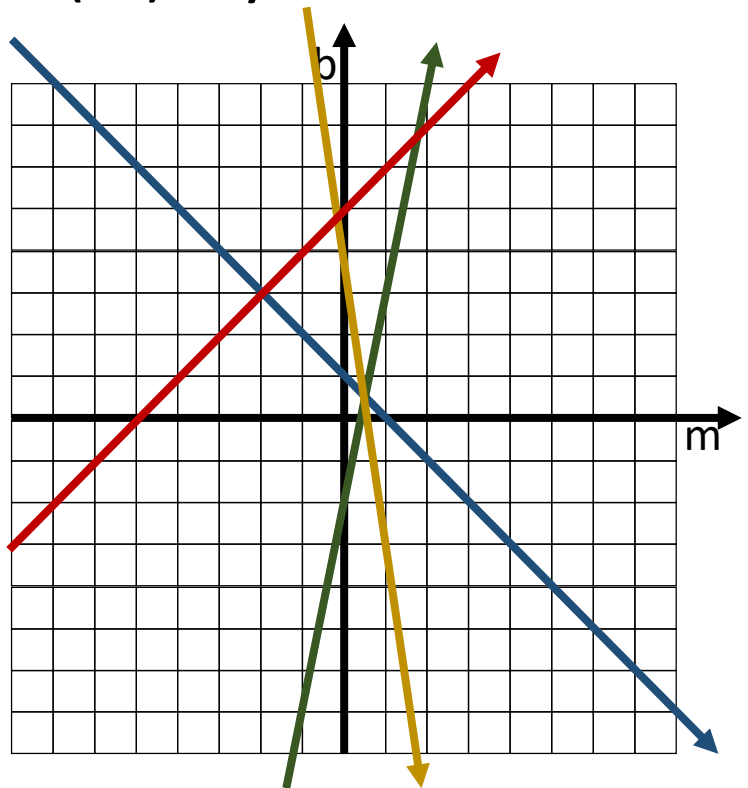
Markus Atkinson

# Hough Transform Basics

- Want to fit your data with some sort of equation: **line**, ellipse, helix, etc..
- y= f(x)=mx+b  <- Hough Transform -> b = f(m) = y-mx

REAL SPACE



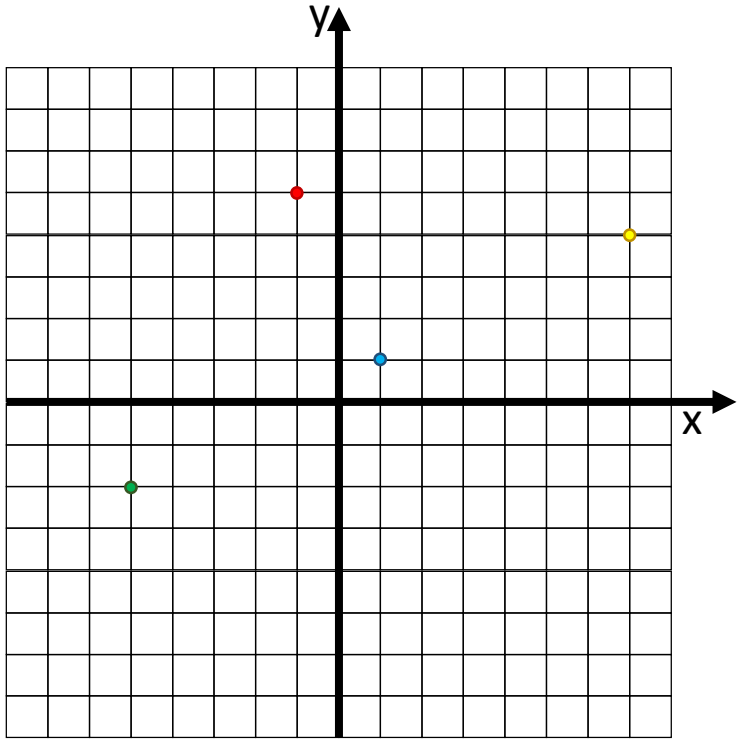



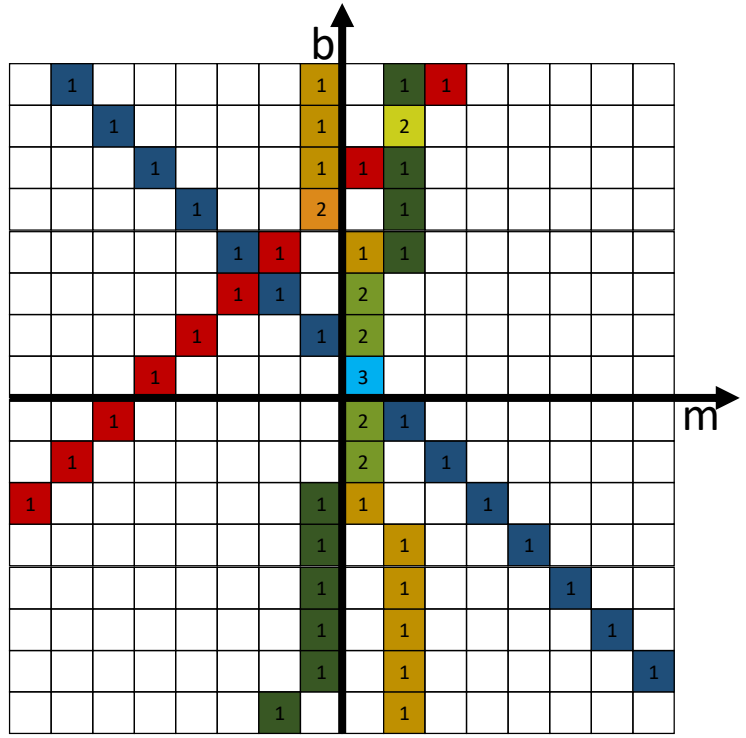

HOUGH SPACE

# Graph Nodes Vote into Accumulators

- Each node in the graph is effectively voting for pairs of parameters that it likes. Infinitely many of them.
- Discretize Hough space into an accumulator matrix (optimal binning?)

REAL SPACE

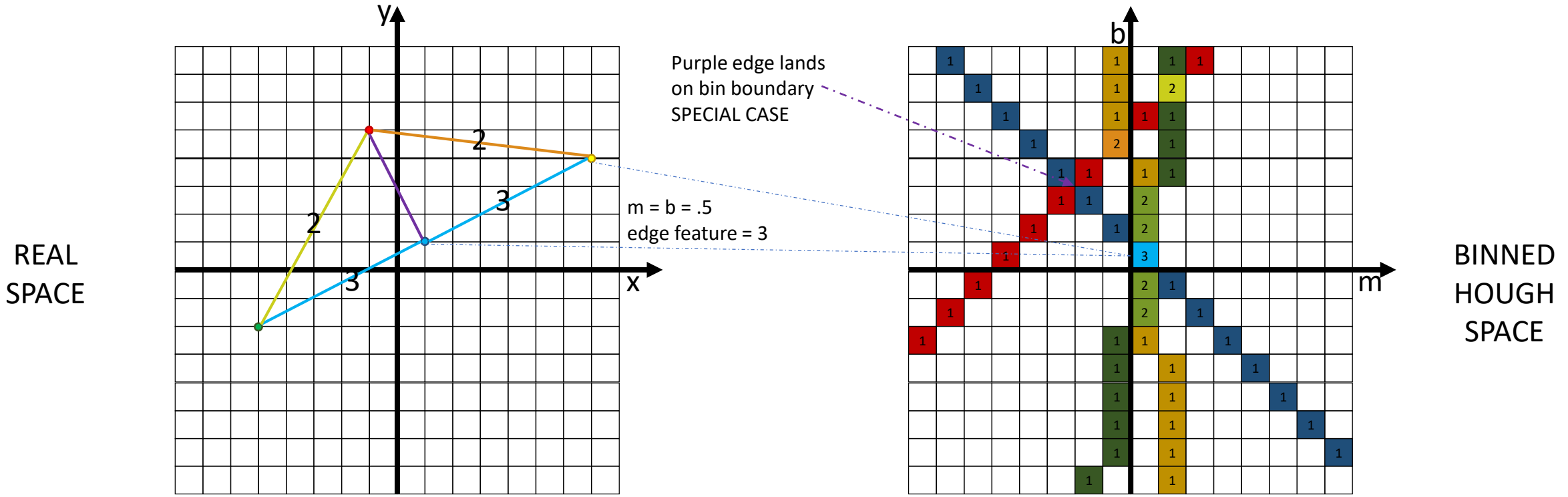



BINNED HOUGH SPACE

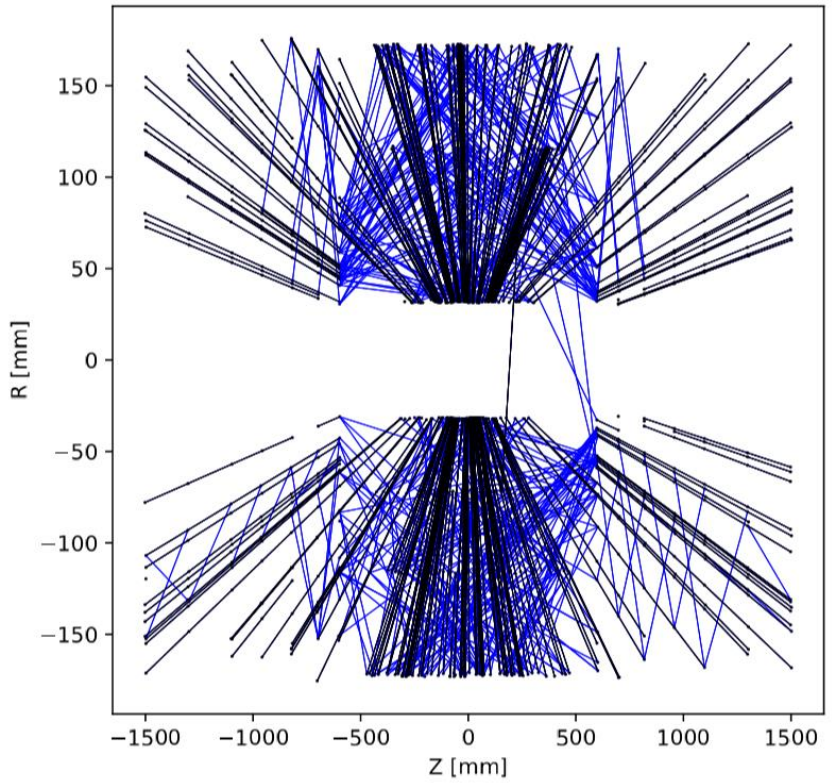
# Extract Edge Feature

- Edge has 2 points, meaning we can easily just calculate m and b
- Use the real m and b of the edge to address the corresponding bin in the accumulator and assign vote count as edge feature



# Orientation matters (RZ plane)

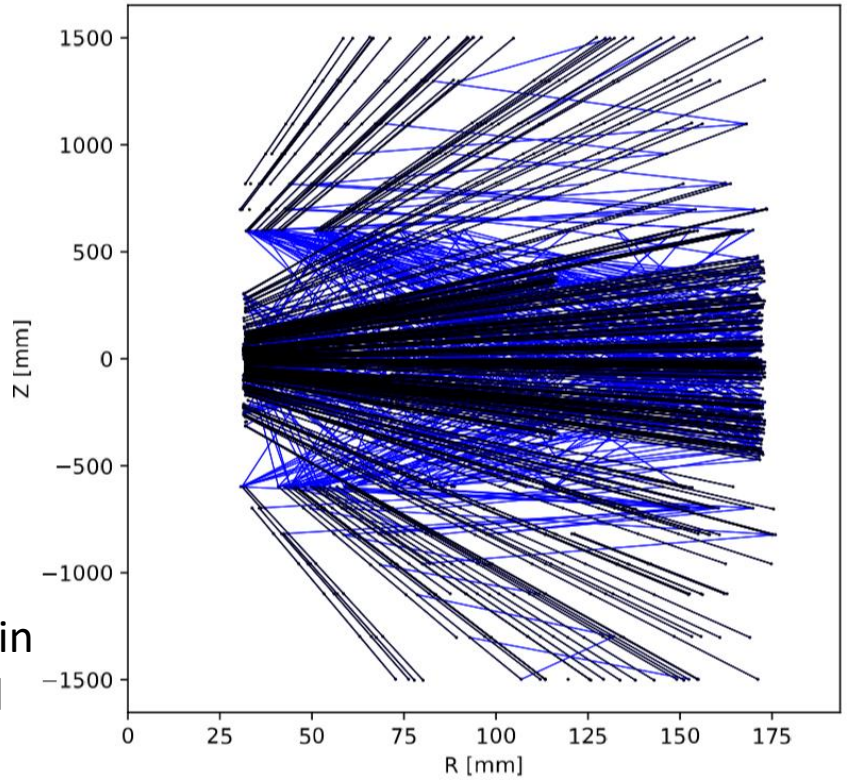
- Vertical lines are an exception we need to avoid, because they do not fit the standard form of a line equation (x = constant, y=mx+b)




Lots of vertical tracks in barrel in this orientation, means range of slope(m) needs to be large

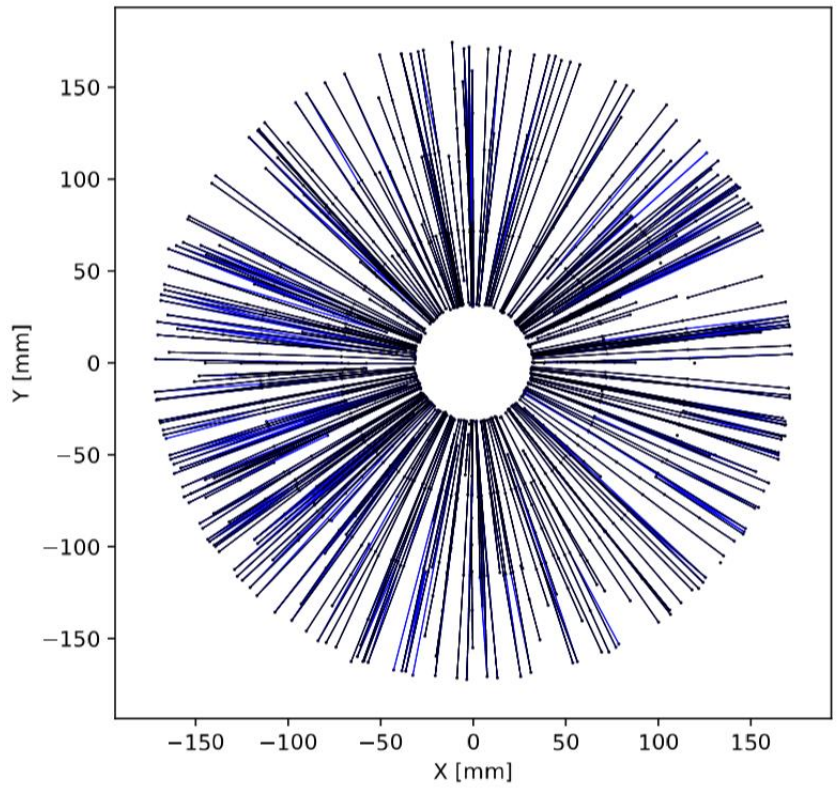
AVOID VERTICAL EDGES

Inverting R and Z removes vertical edges, range of slope in accumulator can be truncated



# Orientation matters (RPhi plane)

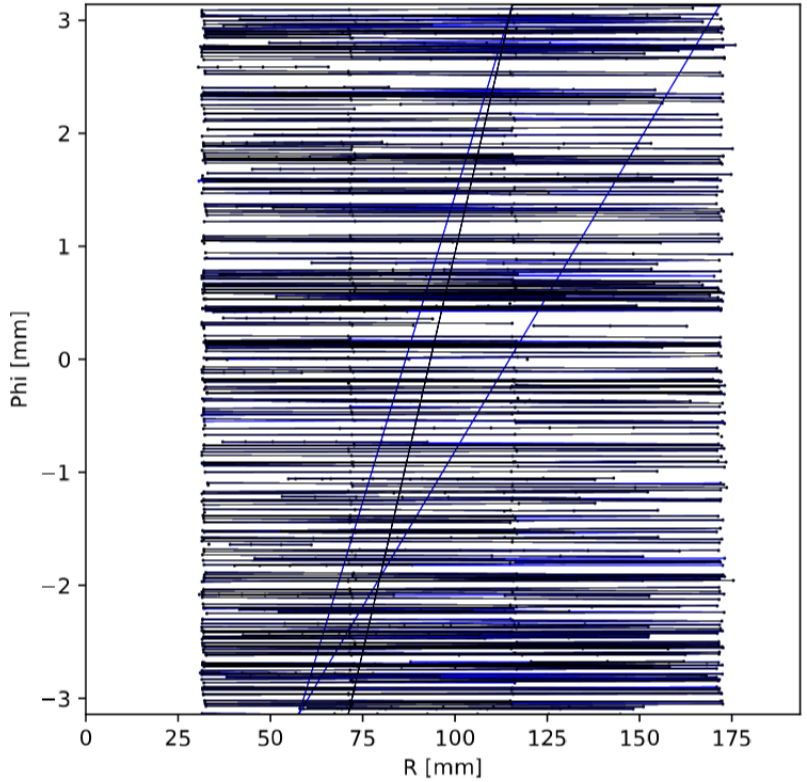
- Vertical lines are an exception we need to avoid, because they do not fit the standard form of a line equation (x = constant, y=mx+b)



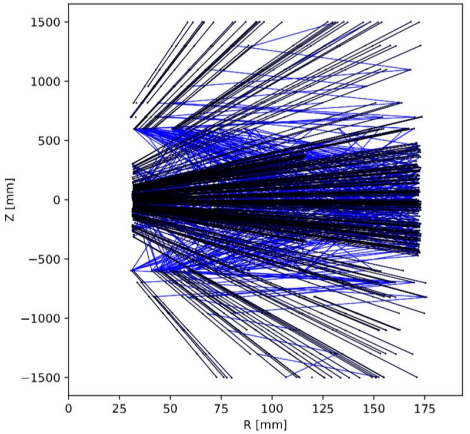

XY plane has further complications. Vertical tracks and tracks from opposite sides of detector have similar parameters

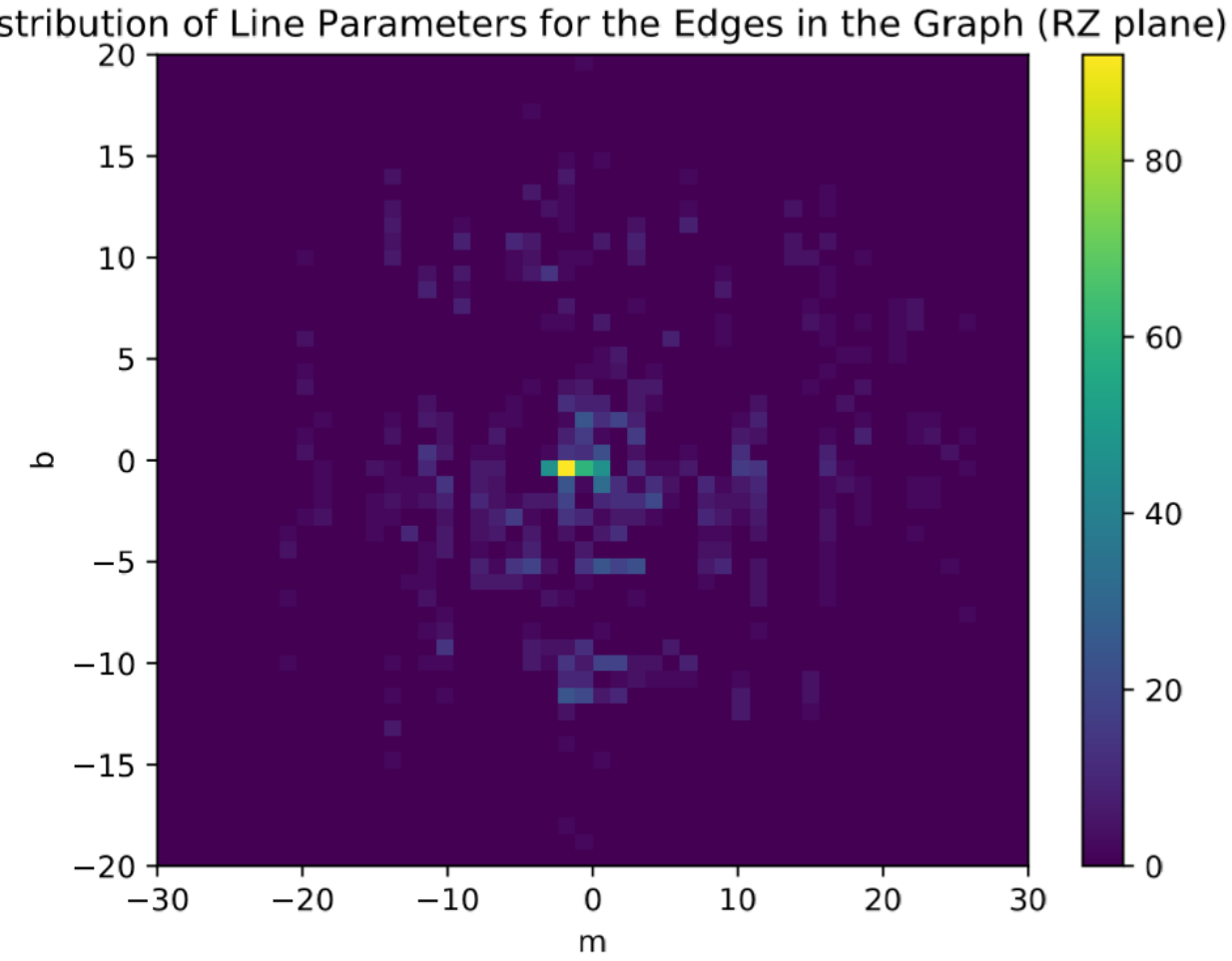
AVOID VERTICAL EDGES

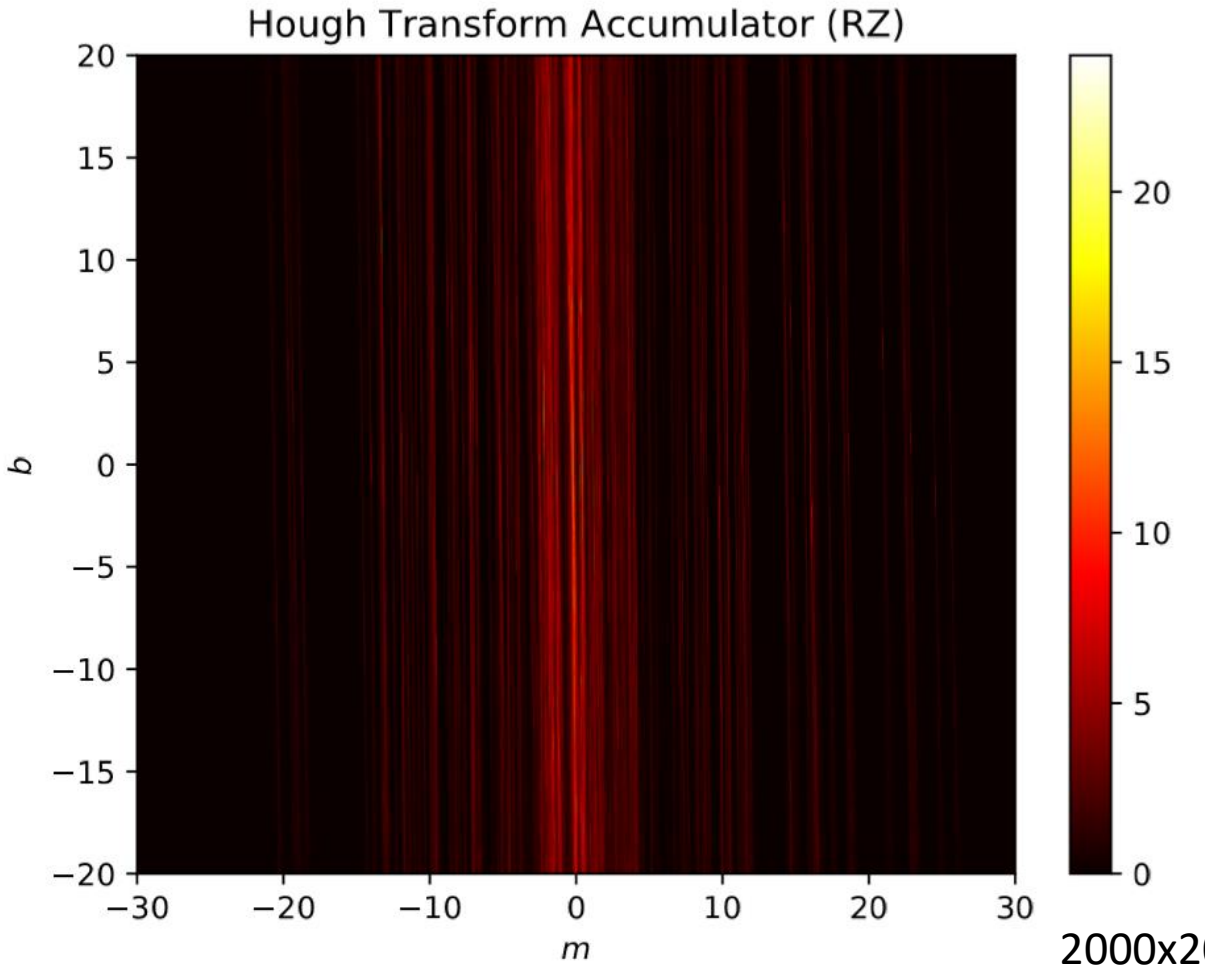
All tracks are horizontal in the Rphi plane



# Region of Interest (RZ plane)



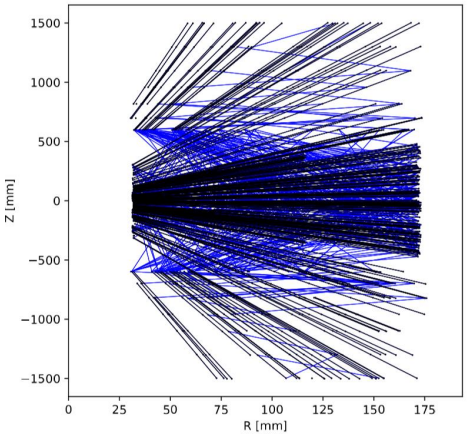



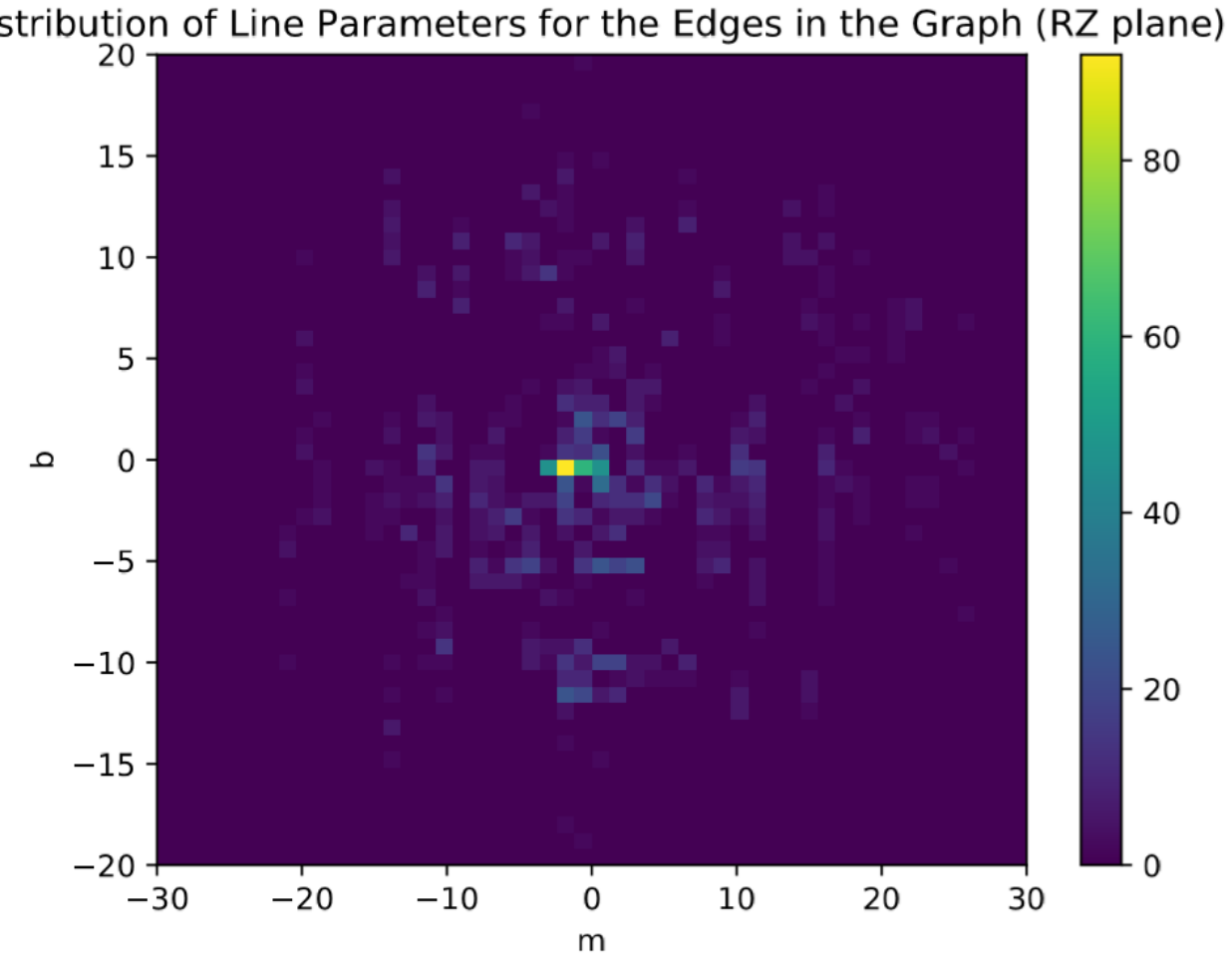



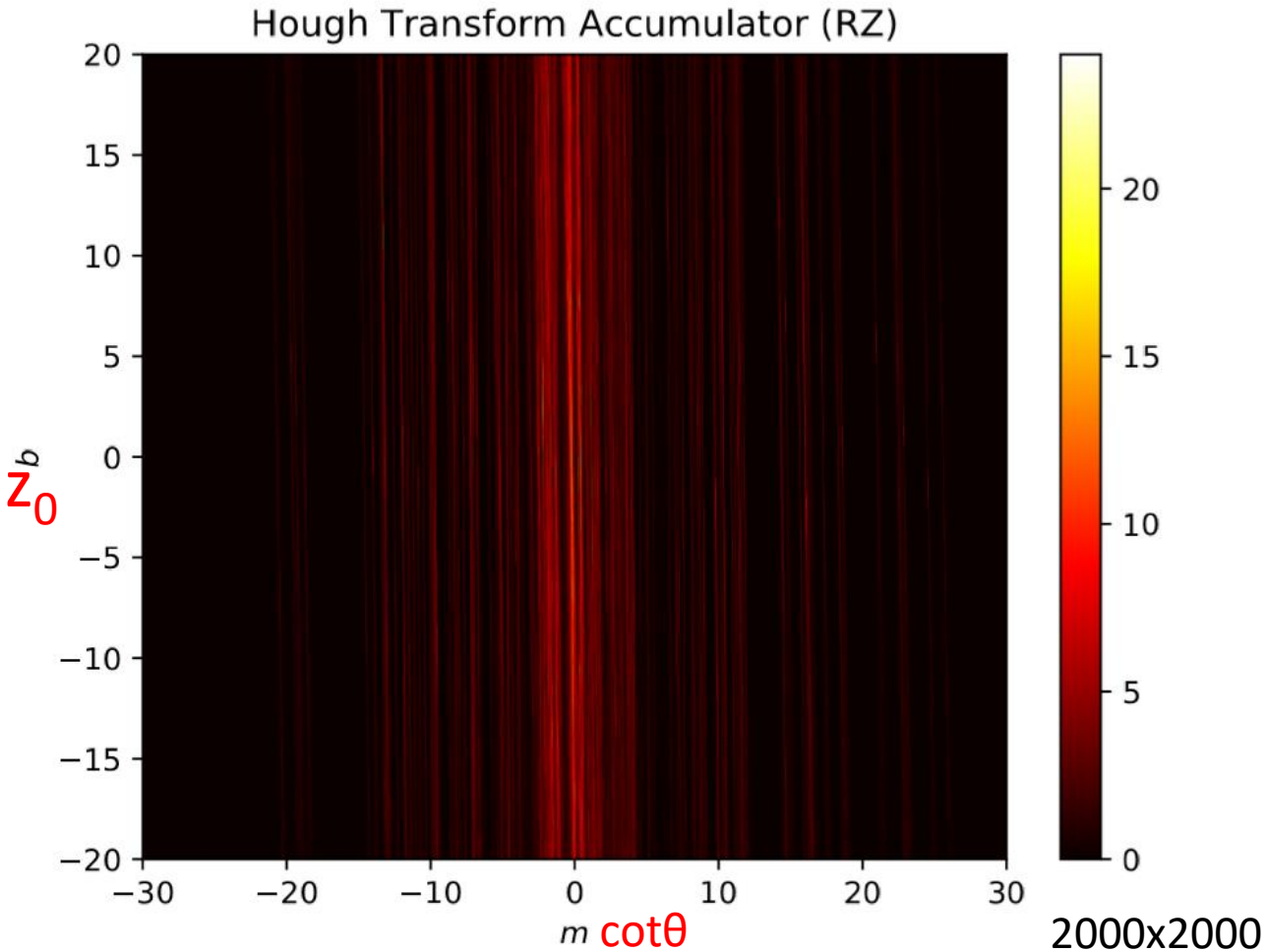

2000x2000

# Region of Interest (RZ plane)

$$z = z_0 + 2\rho \sin^{-1}\left(\frac{R}{2\rho}\right)\cot\theta \longrightarrow z_0 + R\cot\theta$$

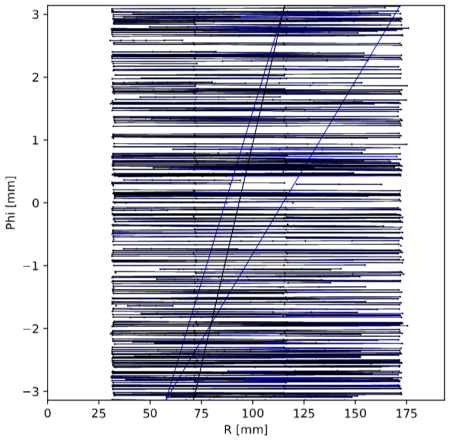


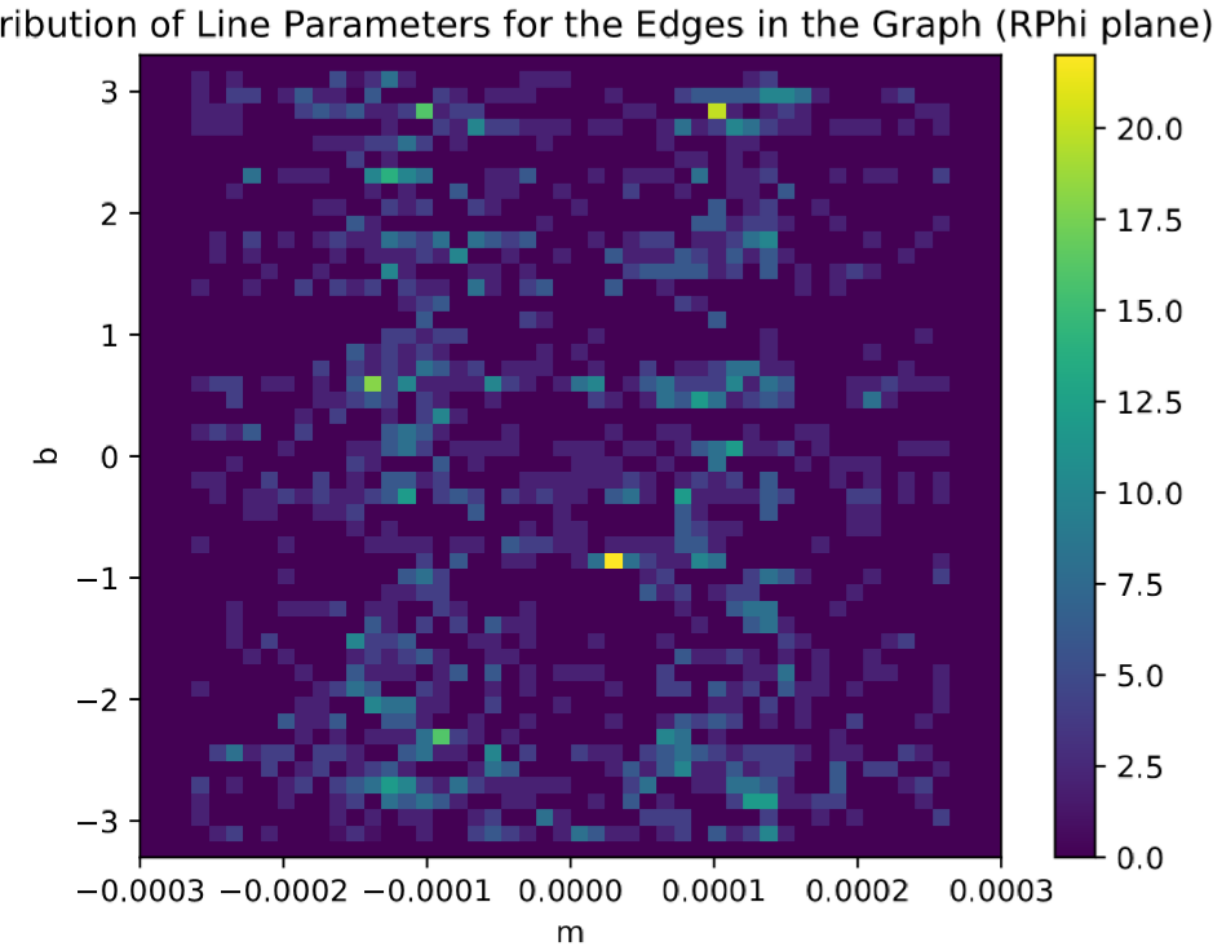



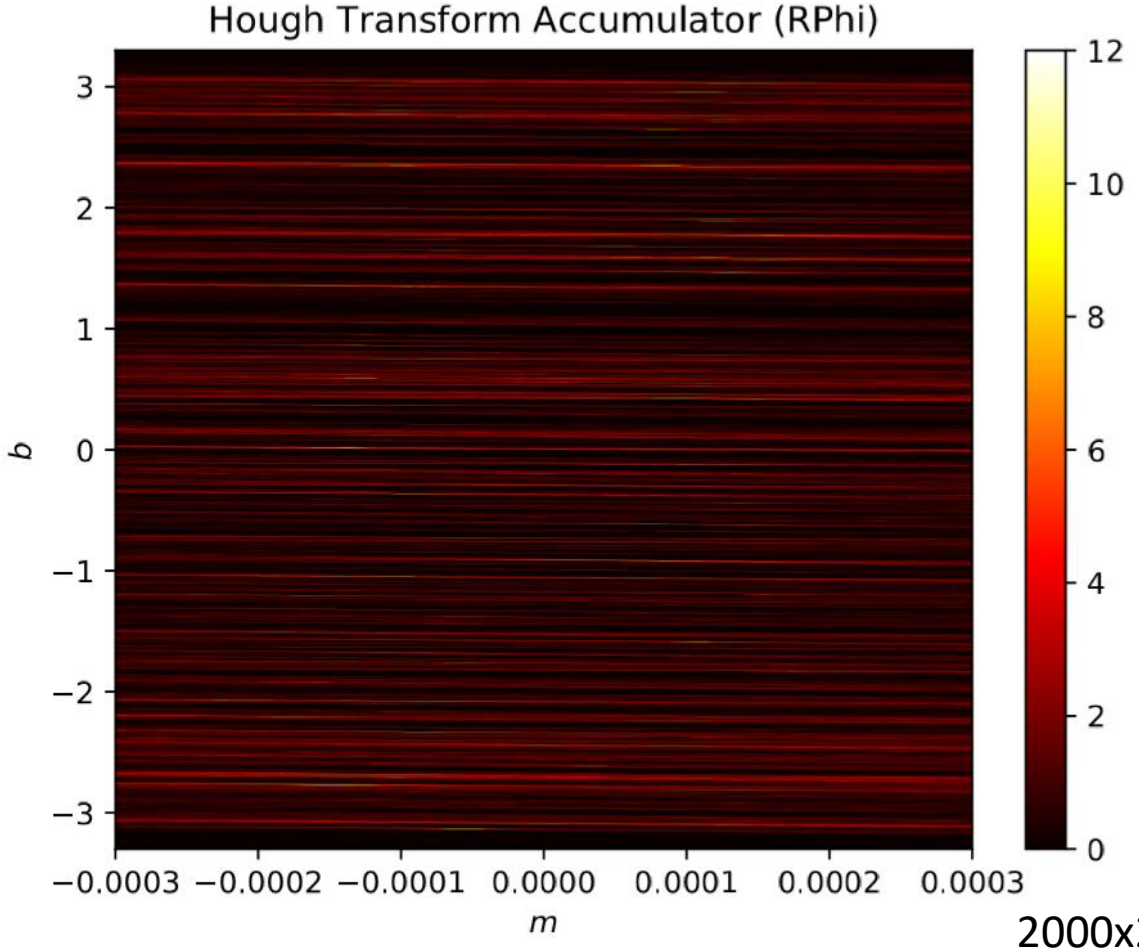




2000x2000

# Region of Interest (RPhi plane)



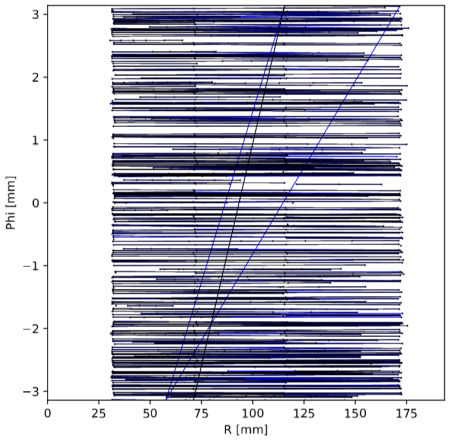



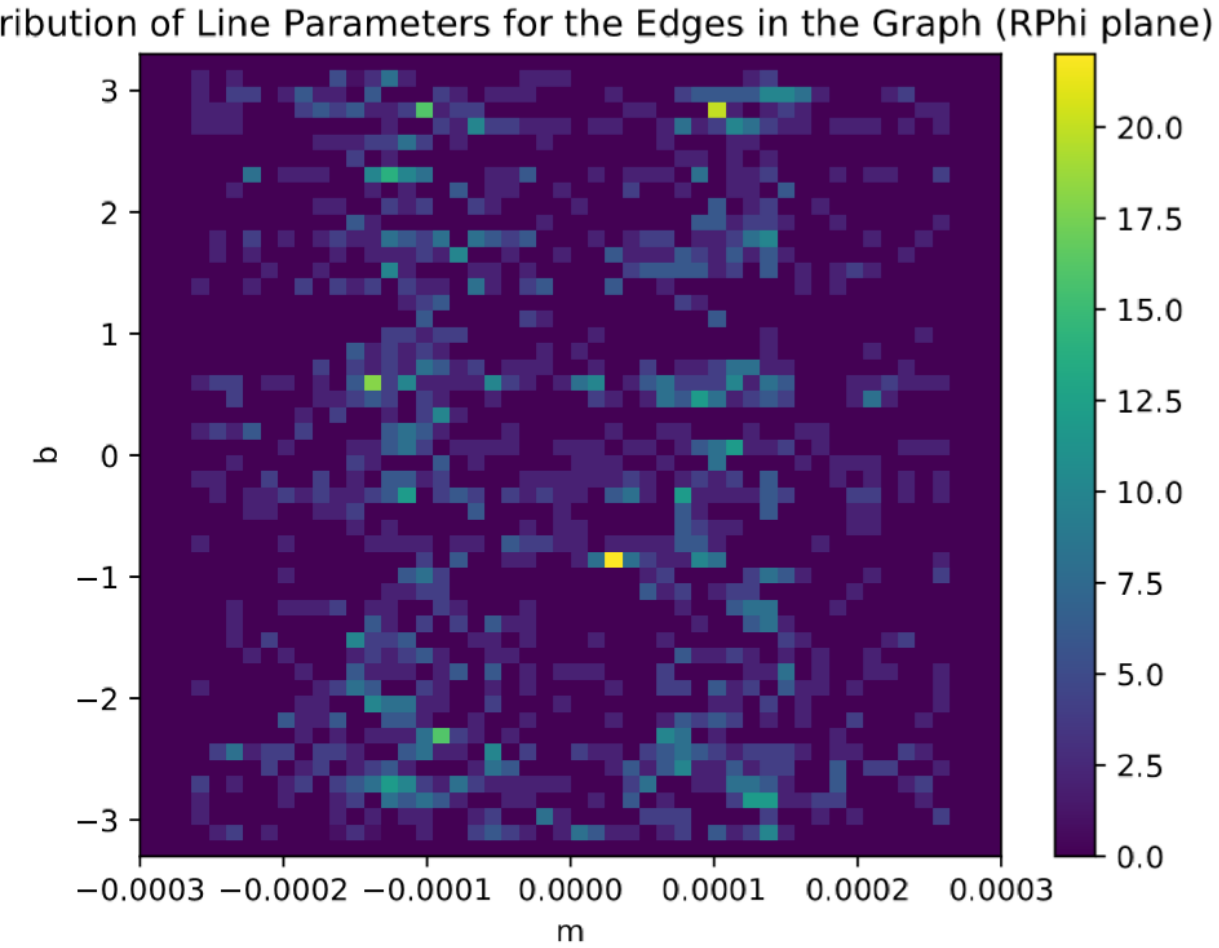



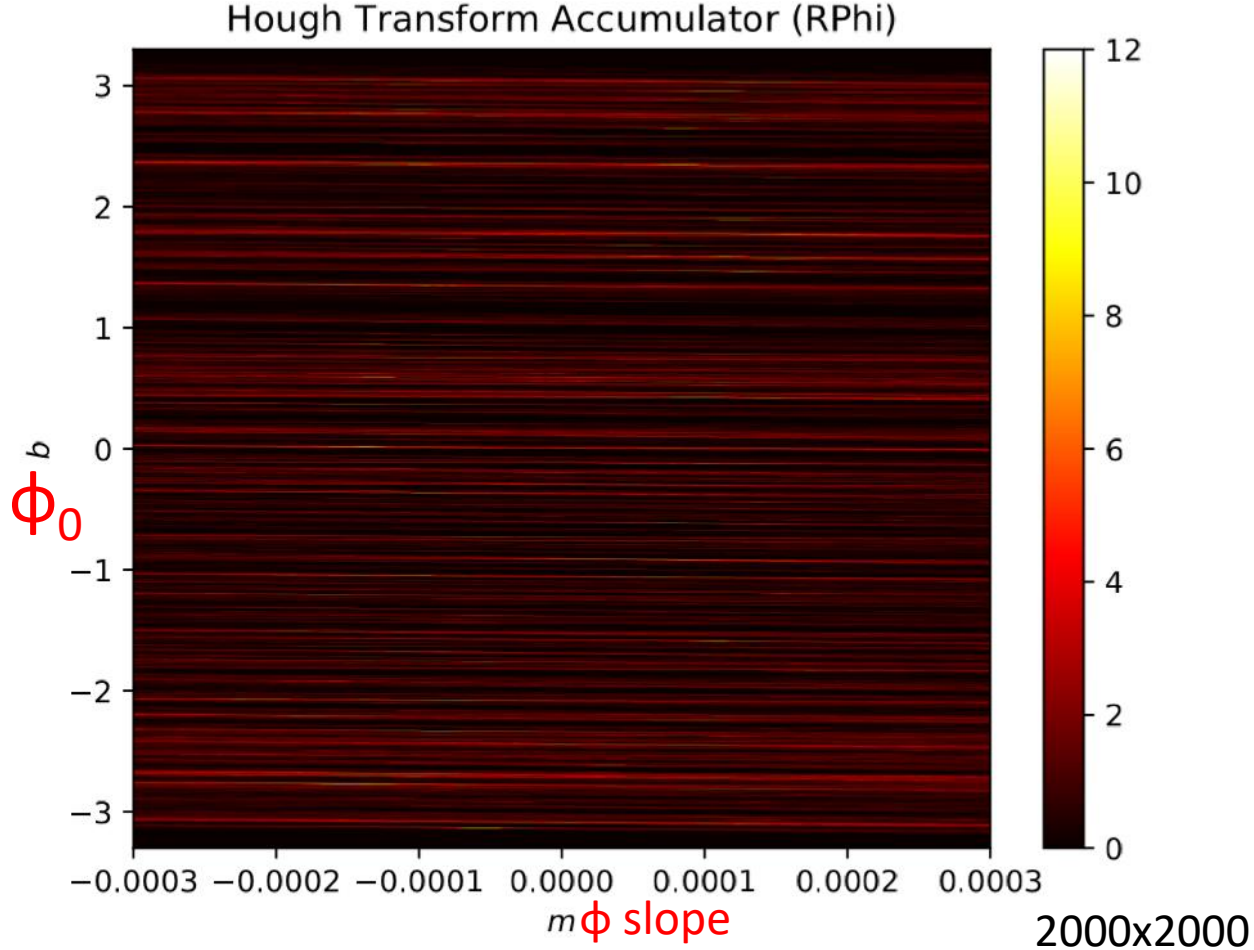

2000x2000

# Region of Interest (RPhi plane)

$$\phi = \phi_0 - \sin^{-1}\left(\frac{R}{2\rho}\right) \rightarrow \phi_0 - \frac{R}{2\rho}$$




$\Phi_0$

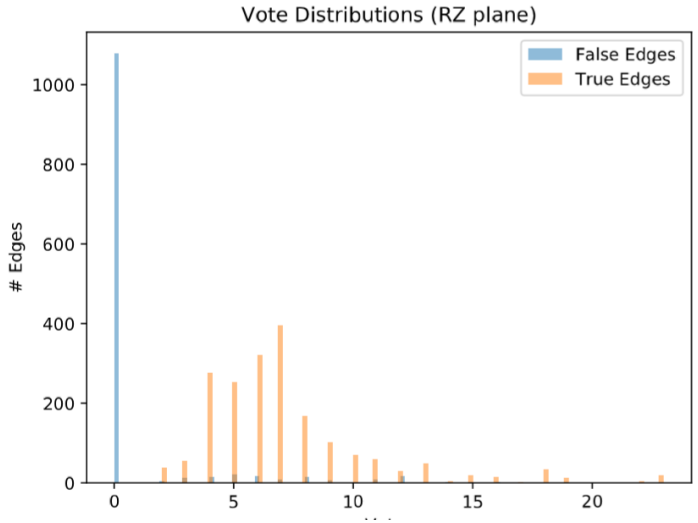
$m\Phi$ slope

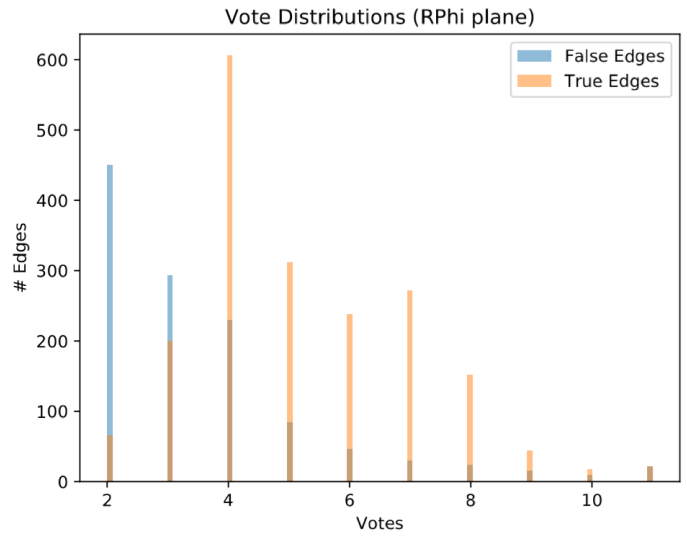
2000x2000

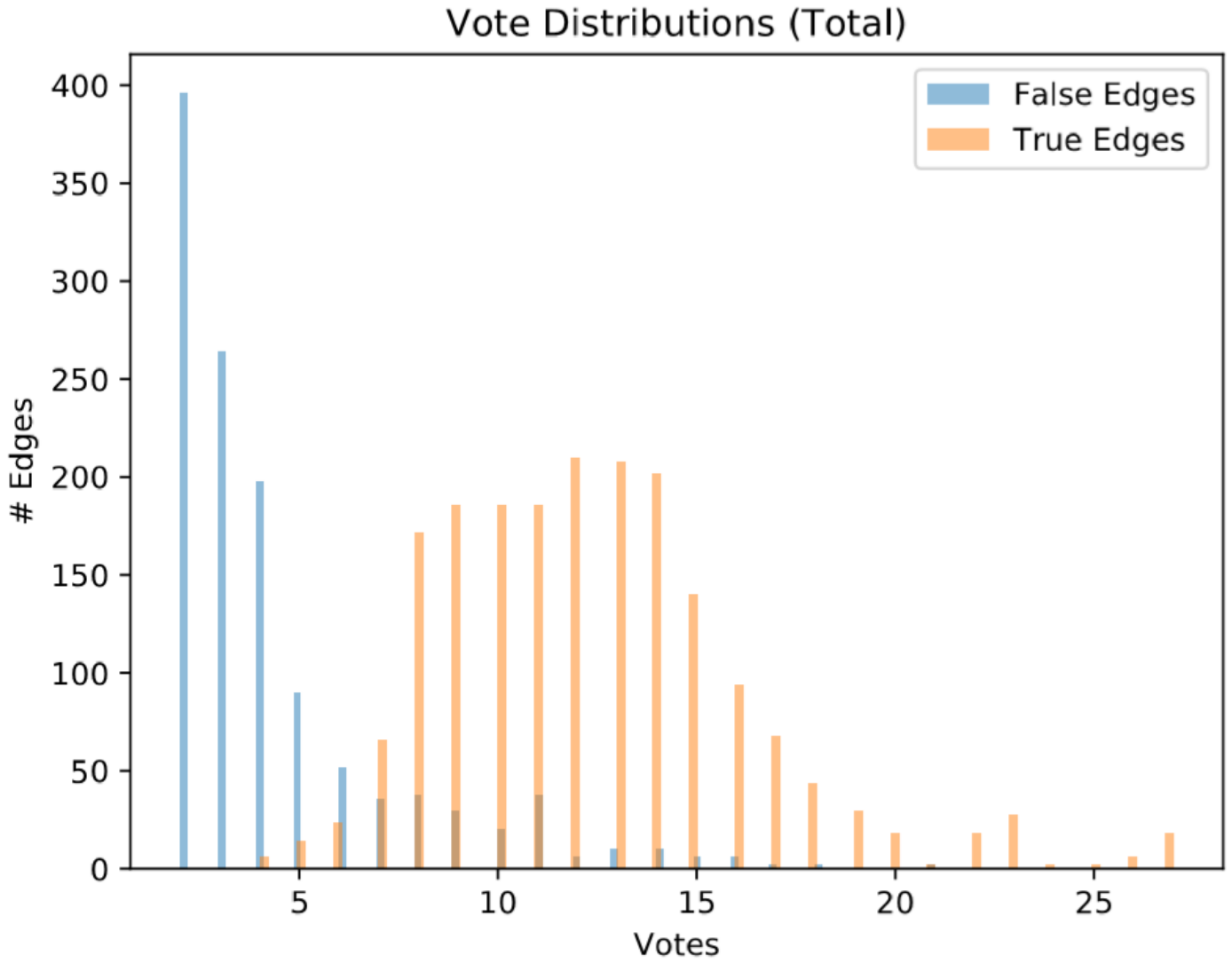
# Accumulator Window Cuts

- By requiring an edge fall inside the accumulator window, you are enforcing 4 geometric cuts on the edge. Luckily these are cuts we already make
  - RZ plane
    - The b parameter here corresponds to z0 (cut we already make)
    - The m parameter here corresponds to z-slope (related to eta, a cut we already make)
      - $\eta = -\ln\left(-m + \sqrt{m^2 + 1}\right) = -\ln\left(\tan\frac{\theta}{2}\right), \qquad m = \cot\theta$
  - Rphi plane
    - The b parameter here corresponds to phi0
      - No cut is made here, accumulator covers [-pi, pi]
    - The m parameter here corresponds to the phi-slope (another cut we already make)
      - $m = \frac{-1}{2\rho}, \quad \rho = \frac{p_T}{qA}$

# Vote Extractions



# Use the votes to cut edges?

- Edges are defined by 2 nodes, so they will always have atleast 2 votes if they land inside the accumulator window
- Requiring a vote of 2 from each accumulator forces the edge to be inside the accumulator window. Thus enforcing the geometric cuts we already do.
- Requiring a vote of 3 though from atleast 1 of the accumulators, means you require the edge to have triplet potential in atleast one of these 2 projections
  - This could potentially be a very powerful cut

# Edge Classifier 2 ($p_T$ > 2 GeV)

- Hough Transform was implemented in pytorch geometric
  - Edge Classifier 2 was easily modified to allow data.edge_attr passing

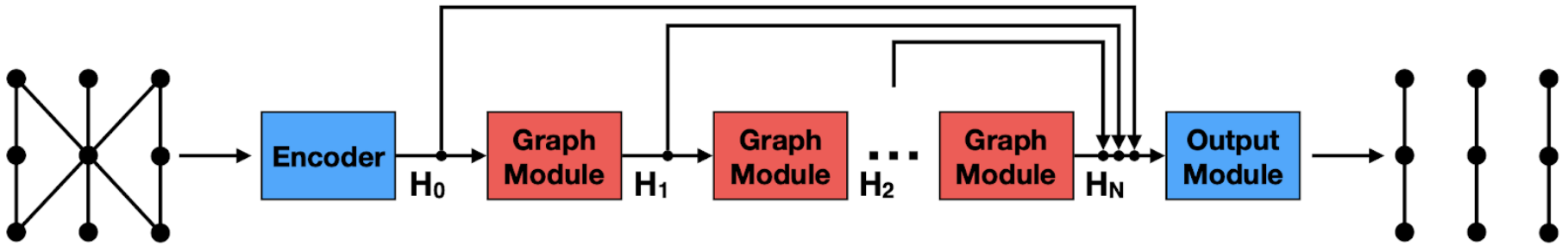
Confusion Matrix without Hough

| | |
|---|---|
| **.999581** | .001837 |
| .000419 | **.998173** |

Confusion Matrix with Hough

| | |
|---|---|
| **.999440** | .000372 |
| .000560 | **.999628** |

| | Input Truth Graphs Track Efficiency | GNN Inferenced Graphs Track Efficiency | Ratio | Fake Fraction |
|---|---|---|---|---|
| No Hough | .946276+-.019355 | .942320+-.019378 | .995819 | .001010+-.002090 |
| With Hough | .946276+-.019355 | **.945112+-.019318** | **.998770** | **.000771+-.001577** |



# To Do

- Further optimize code (make it more pythonic)
  - Have made some major improvements to the algorithm that greatly increased build time, but I think it could be optimized further but it's a bit beyond my level of python at this point
  - Now that the first results look promising this is more important than ever.
  - This increase in performance comes at a large increase in graph construction time
    - Interesting idea. Existing Hough Transform firmware could be adapted for graph construction
- See effect on lower Pt  cuts

# Conclusion

- New edge feature, # nodes that voted for line parameters close to this edge (Will always be 2 or greater)
  - Could also be used to cut edges directly
- Initial run in Edge Classifier 2 gave higher tracking efficiency
  - Results are promising, but are they worth the increased graph construction time?
- What is the optimal binning of the hough space?
  - Too few bins and cant distinguish close edges
  - Too many bins and votes become too sparse in the accumulator
  - Zooming, build multiple accumulators of varying bins and extract multiple vote counts