

GPU-based Online Tracking for the PANDA Experiment

NVIDIA Application Lab Workshop 2014

8 October 2014, Andreas Herten

Outline

- PANDA
 - HEP
 - Our Experiment
 - Online Event Filter
- Algorithms
 - Hough Transform
 - Riemann Track Finder
 - Triplet Finder

- High Energy Physics (*HEP*):

- *HEP Recipe*

1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe

- High Energy Physics (*HEP*):

- *HEP Recipe*

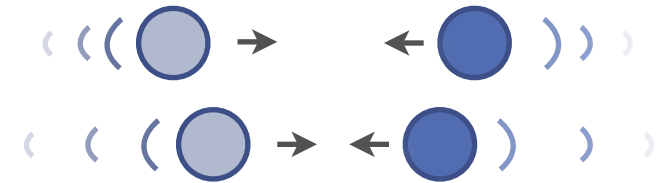
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

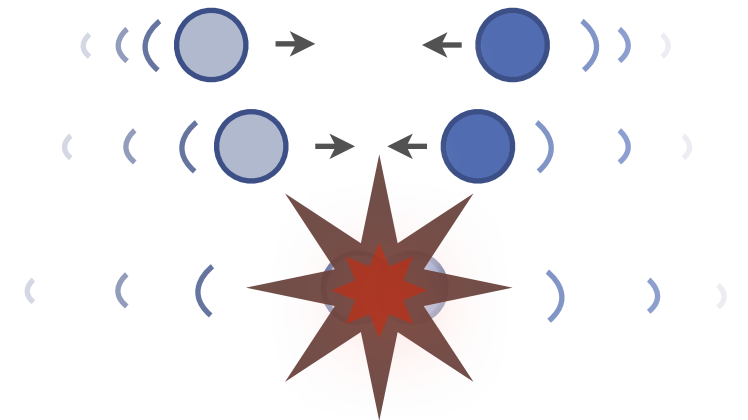
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

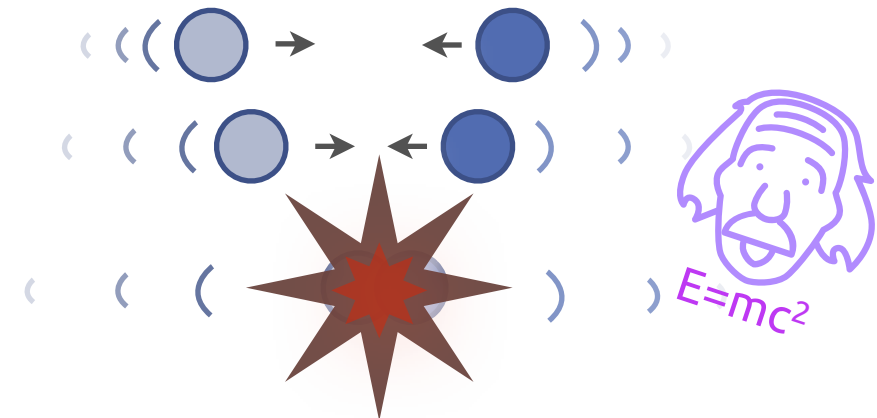
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

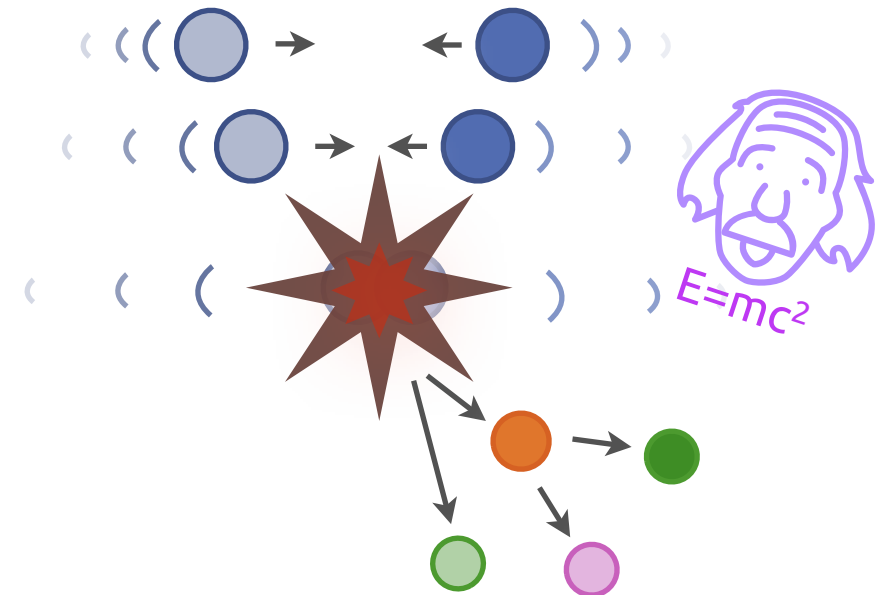
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

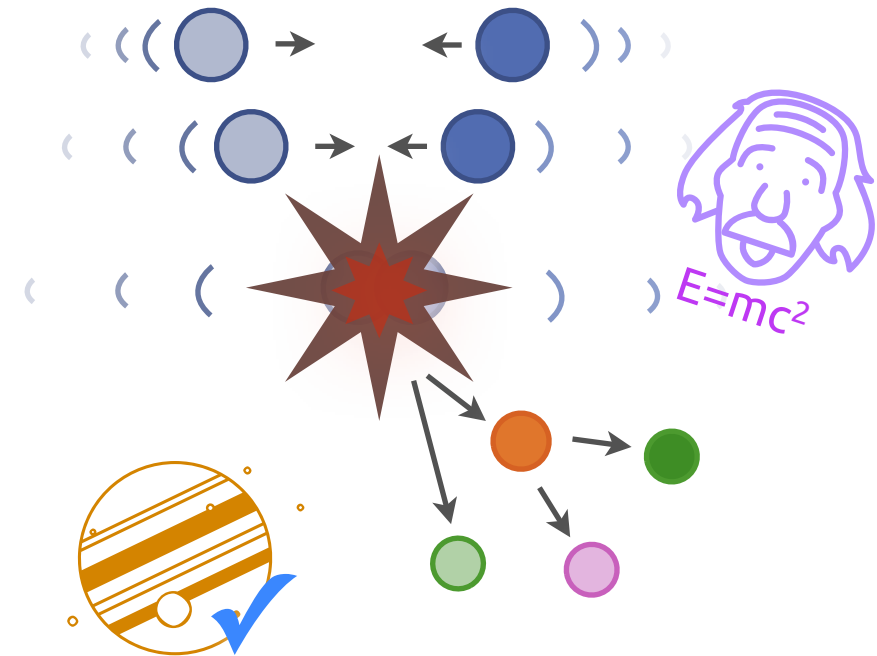
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

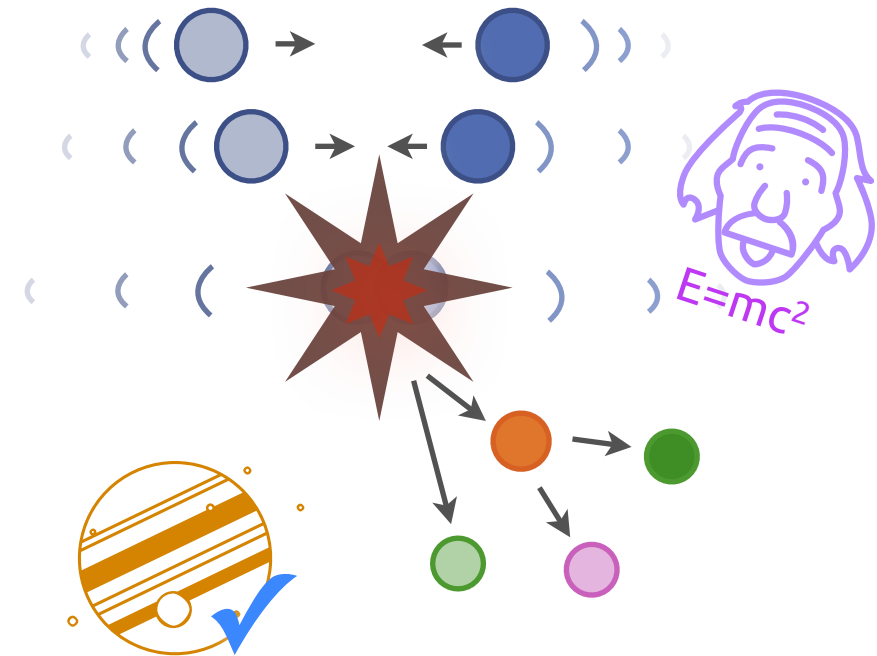
1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



- High Energy Physics (*HEP*):

- *HEP Recipe*

1. Accelerate particles (e, p,...)
2. Accelerate particles more!
3. Smash into each other
4. Look at resulting particles
5. Understand universe



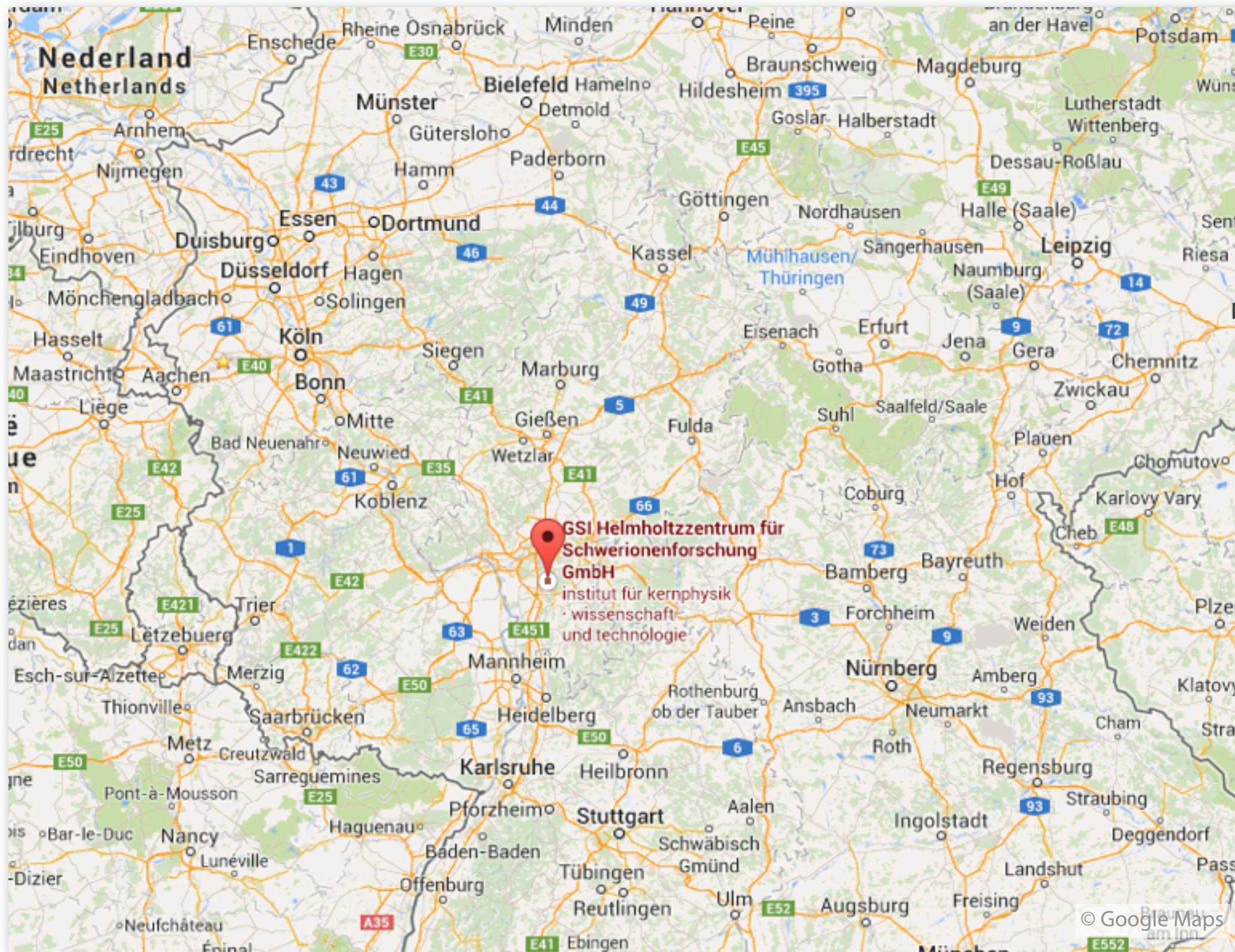
- GPUs are interesting for HEP

- Many events due to high collision rate
- Events **independent**, dividable into subsets
- Many features extractable (**computational intensive**)

- Facility for Antiproton and Ion Research
 - New accelerator complex (Darmstadt, Germany)
 - Next to GSI laboratory
 - Construction in progress, ending 2018
 - Four pillars of research:

APPA	NUSTAR	CBM	PANDA
Atom & plasma physics	Nuclear structure, astro physics	Hadron physics	Hadron physics

- Facility for Antiproton and Ion Research



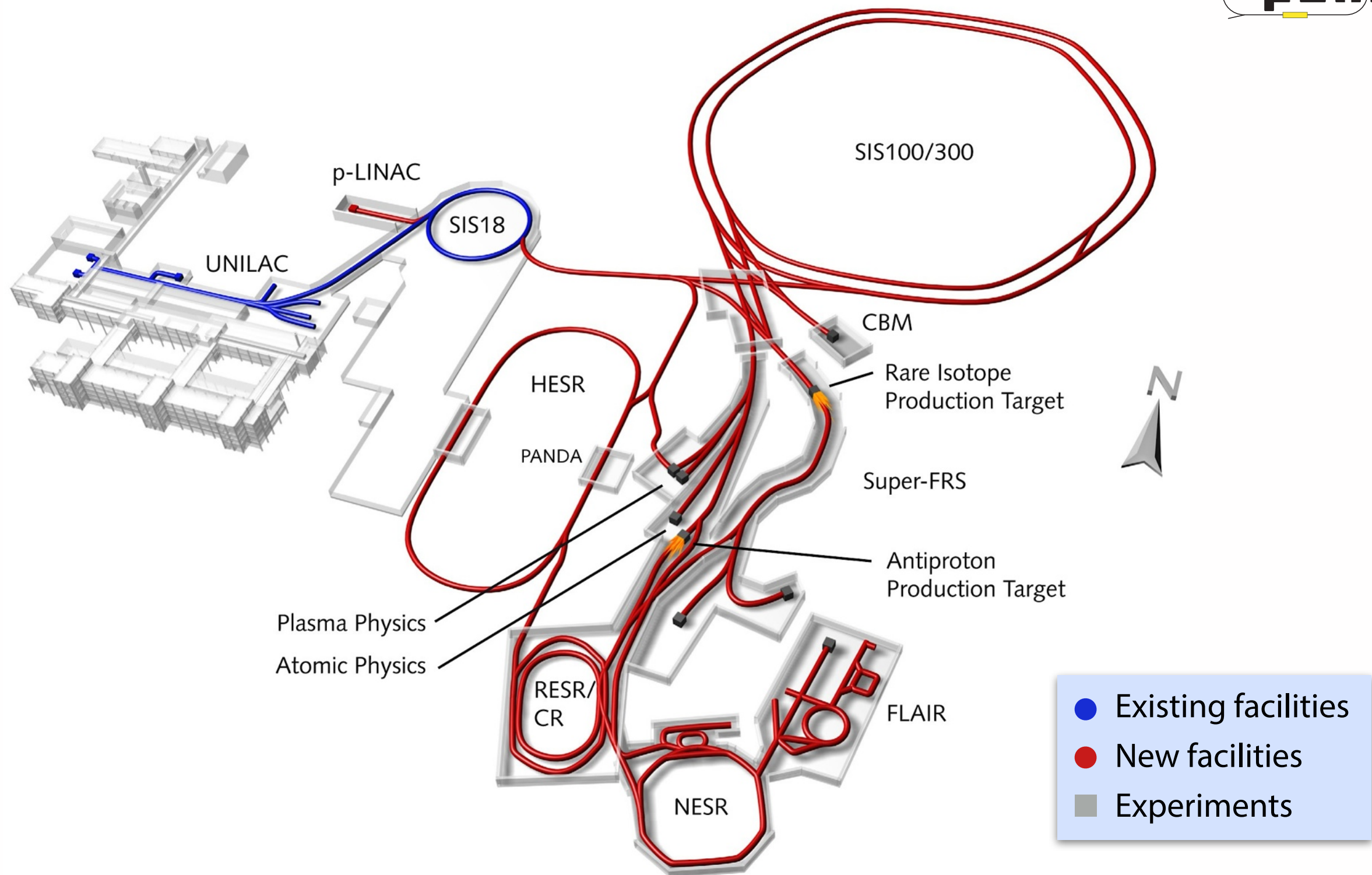
- Facility for Antiproton and Ion Research



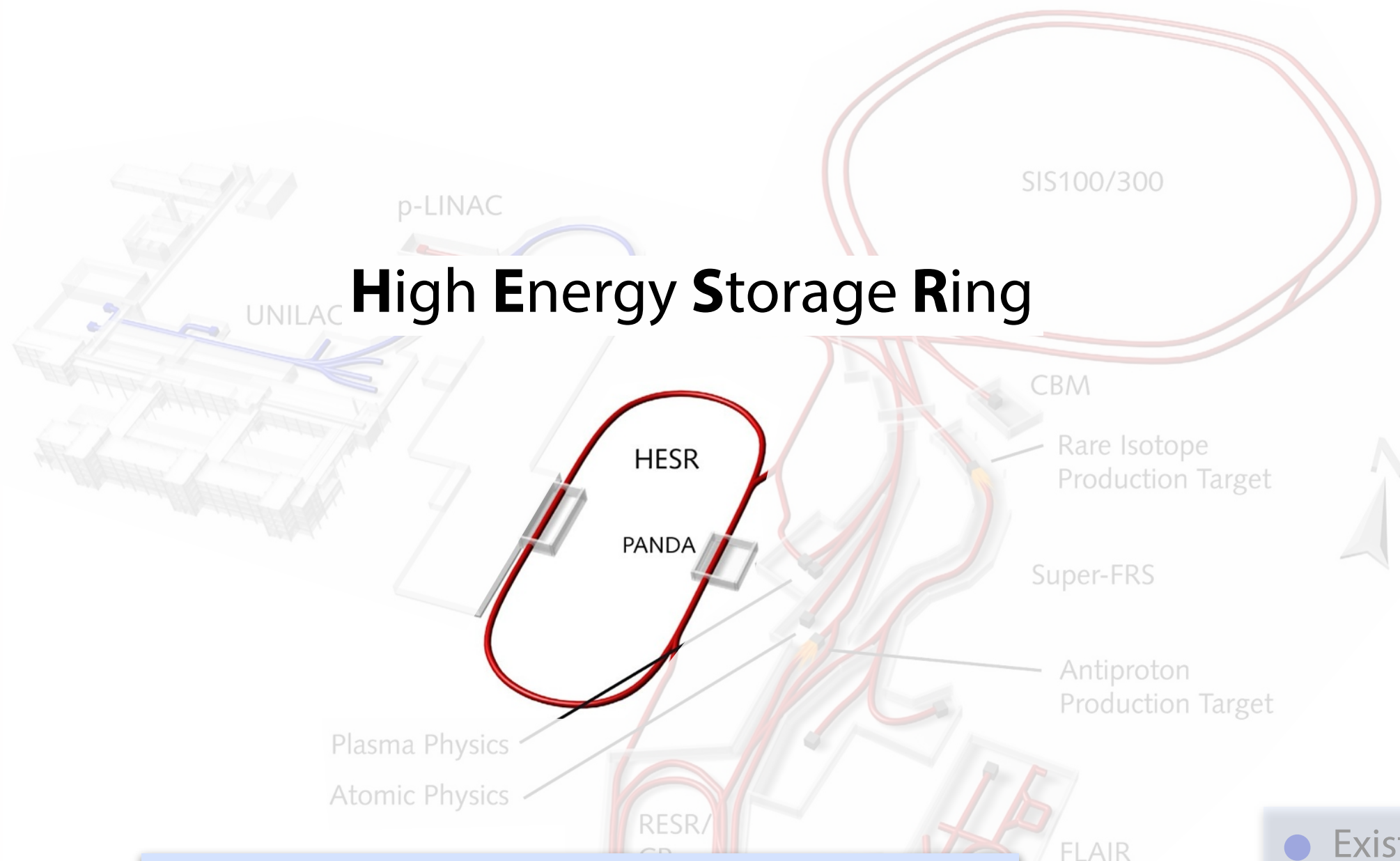
- Facility for Antiproton and Ion Research



FAIR Accelerator Complex



FAIR Accelerator Complex

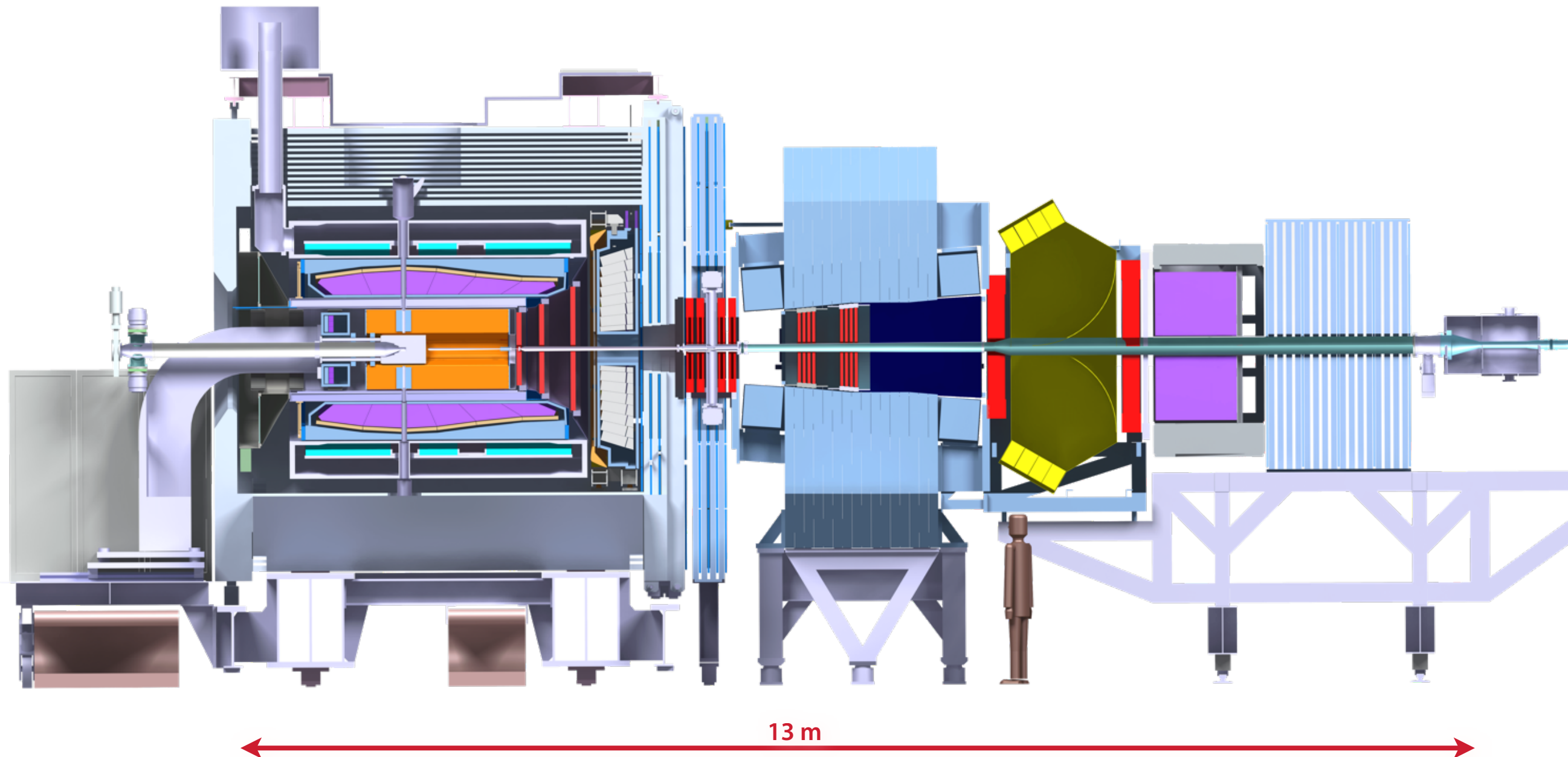


Beam momentum: $p = 1.5 - 15 \text{ GeV}/c$

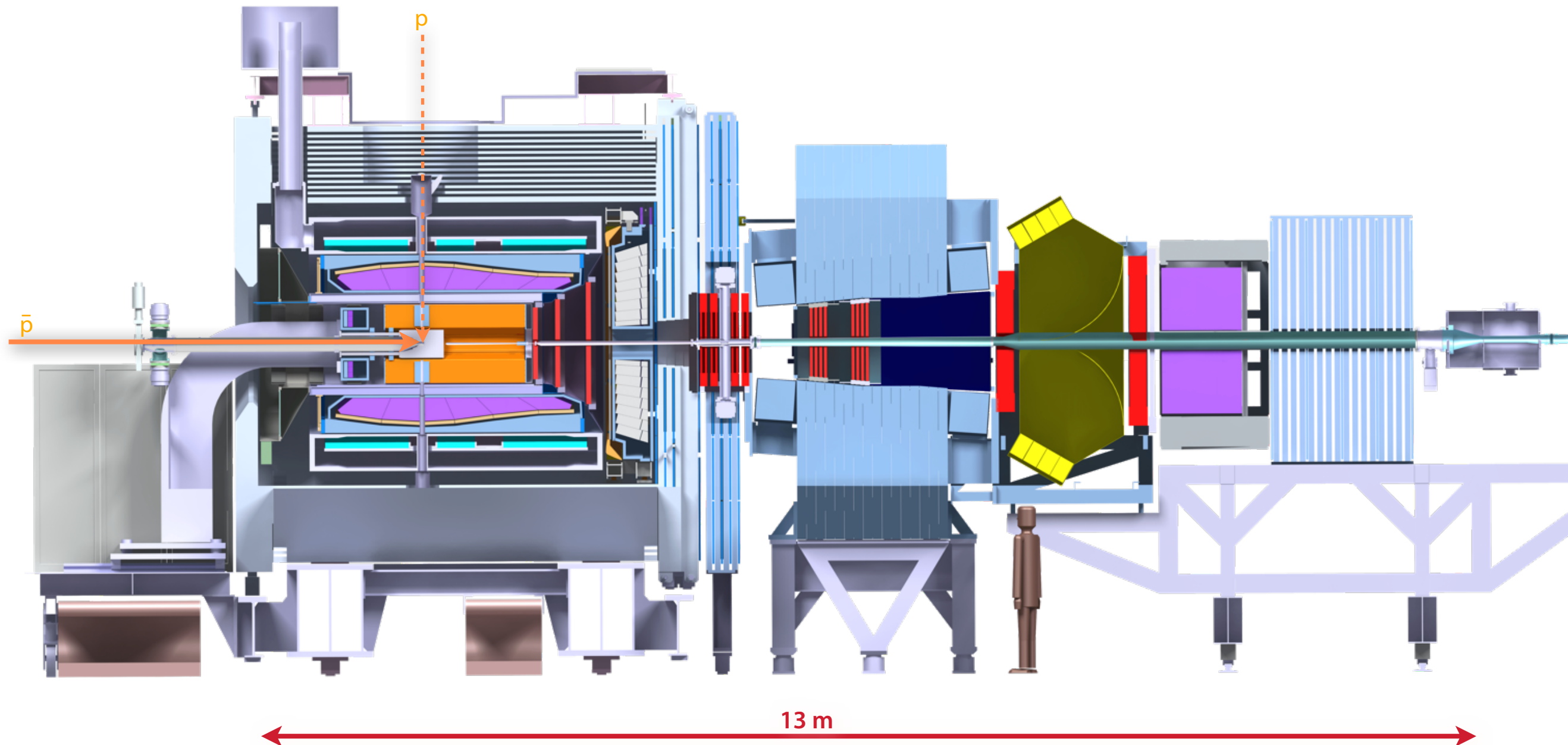
High resolution: $L = 10^{31} \text{ cm}^{-2}\text{s}^{-1}$ $\Delta p/p < 4 \cdot 10^{-5}$

High luminosity: $L = 2 \cdot 10^{32} \text{ cm}^{-2}\text{s}^{-1}$ $\Delta p/p < 10^{-4}$

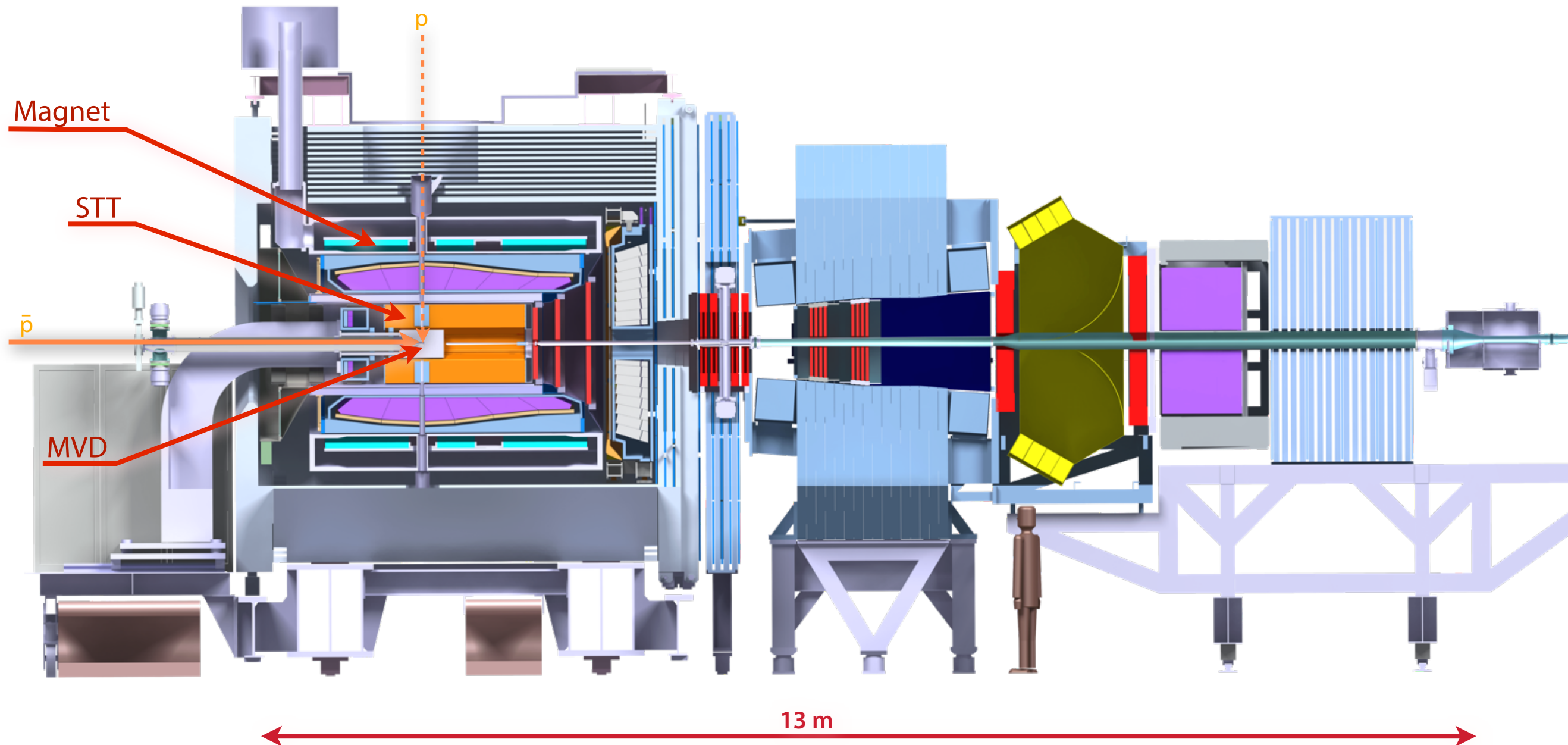
PANDA — The Experiment



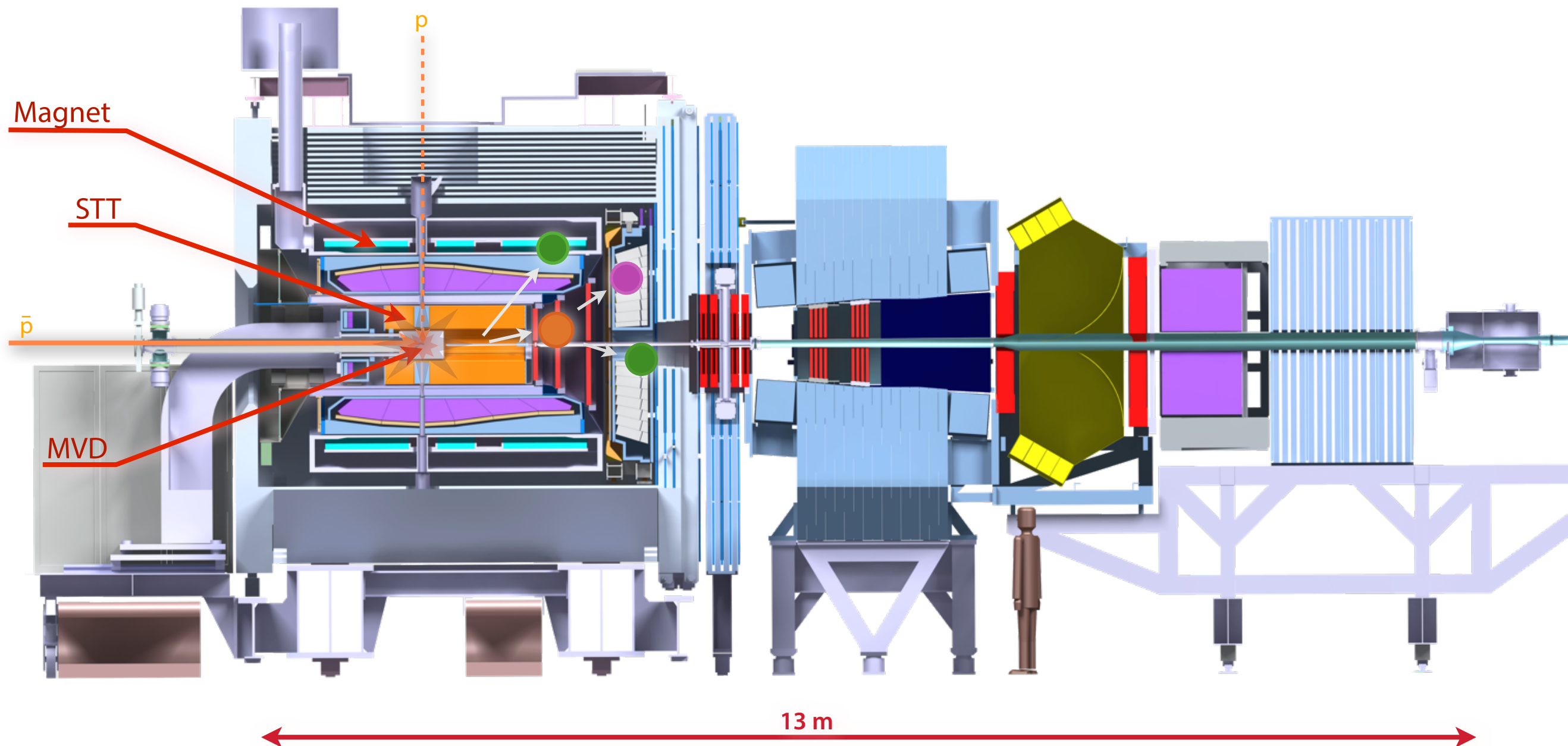
PANDA — The Experiment

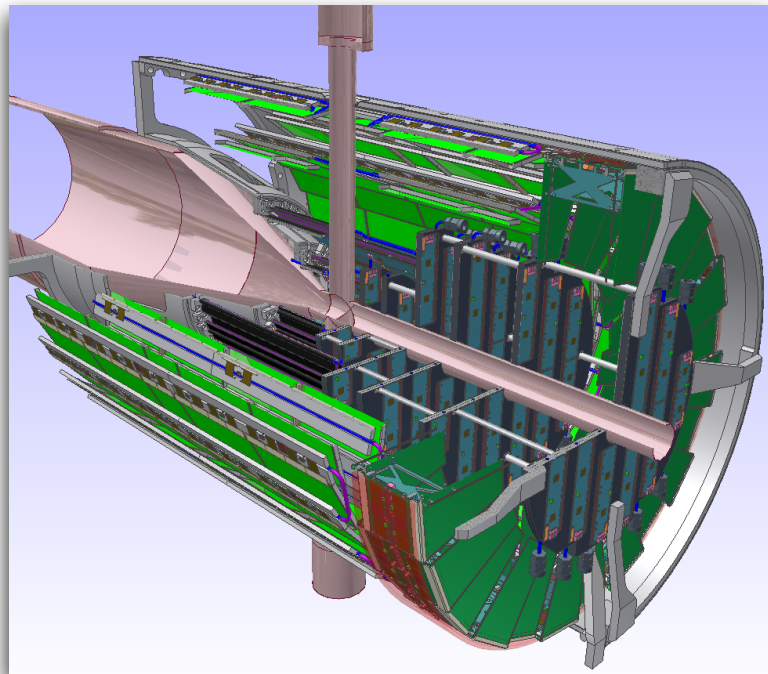


$\bar{P}$ ANDA — The Experiment



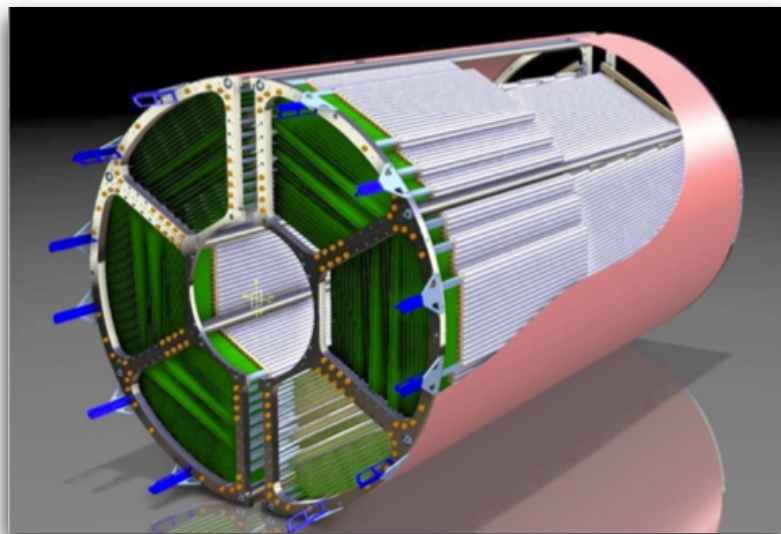
PANDA — The Experiment





Micro Vertex Detector

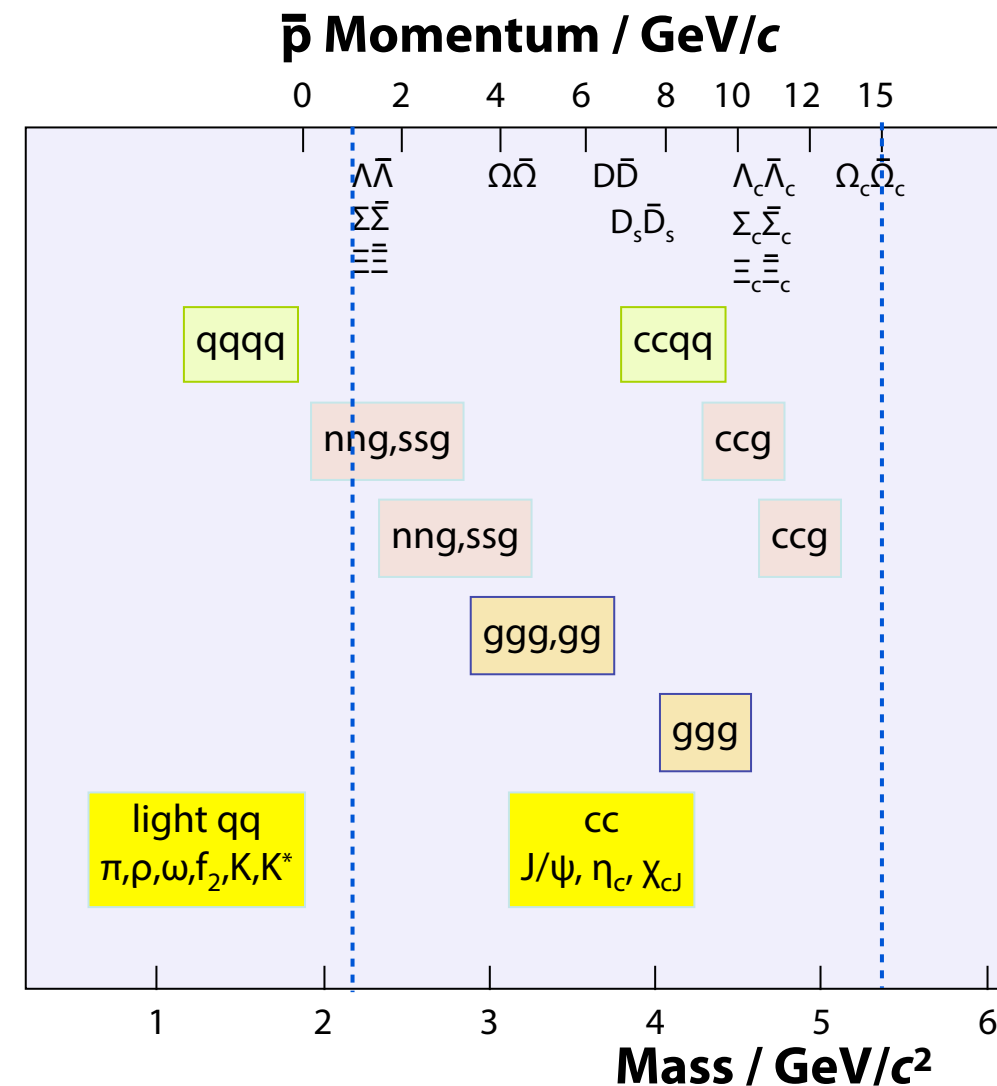
- Silicon-based **pixel** + **strip** detector
- **10 000 000** + **200 000** channels
- Vertex resolution: $< 100 \mu\text{m}$



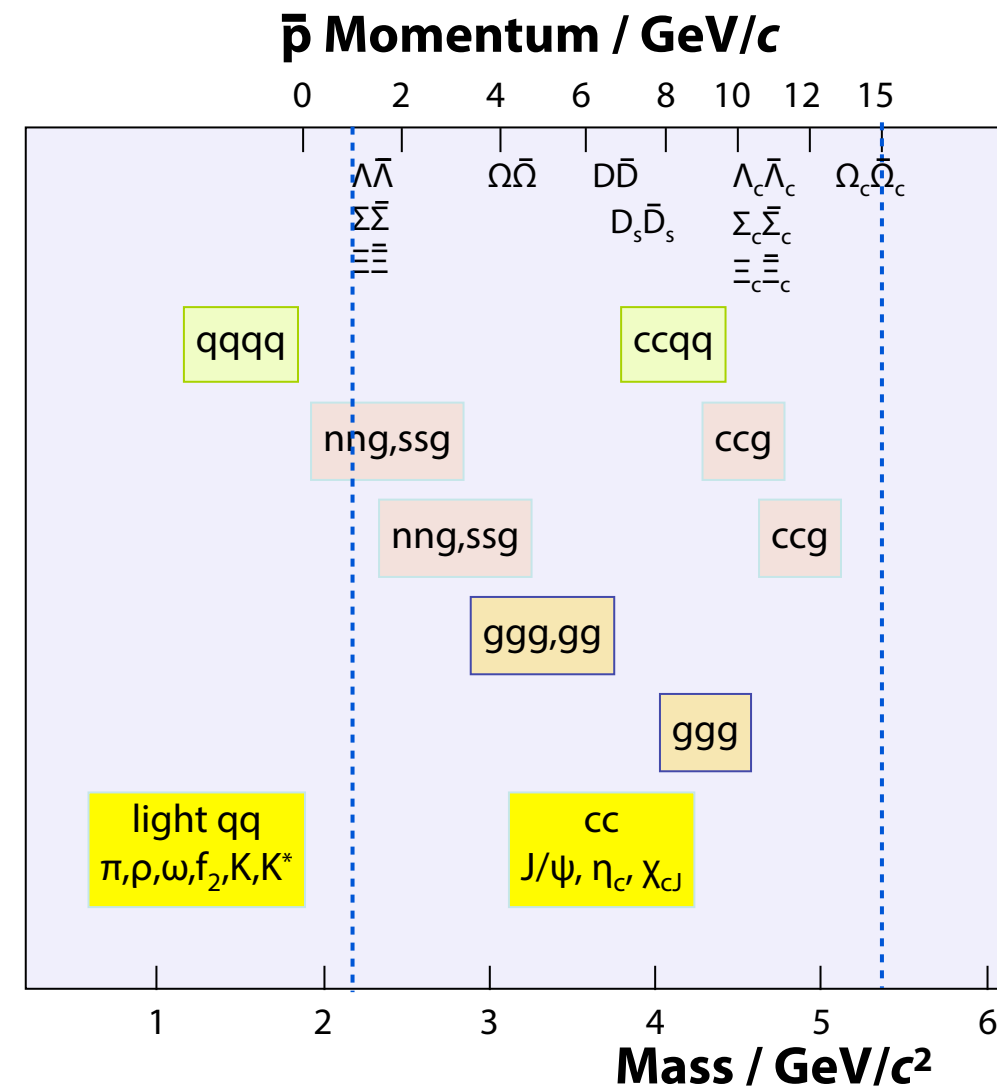
Straw Tube Tracker

- 4636 small drift tubes ($\varnothing 1 \text{ cm}$)
 - Drift times: $< 250 \text{ ns}$
- 26 layers, 8 skewed
- Material budget: 1.2 % radiation length

- Meson spectroscopy
 - Light mesons
 - Charmonium
 - Open charm
 - Exotic states
 - Glueballs
 - Hybrids
 - Molecules/multiquarks
- Baryon production
- Nucleon structure, e.m. processes
- Charm in nuclei
- Strangeness physics



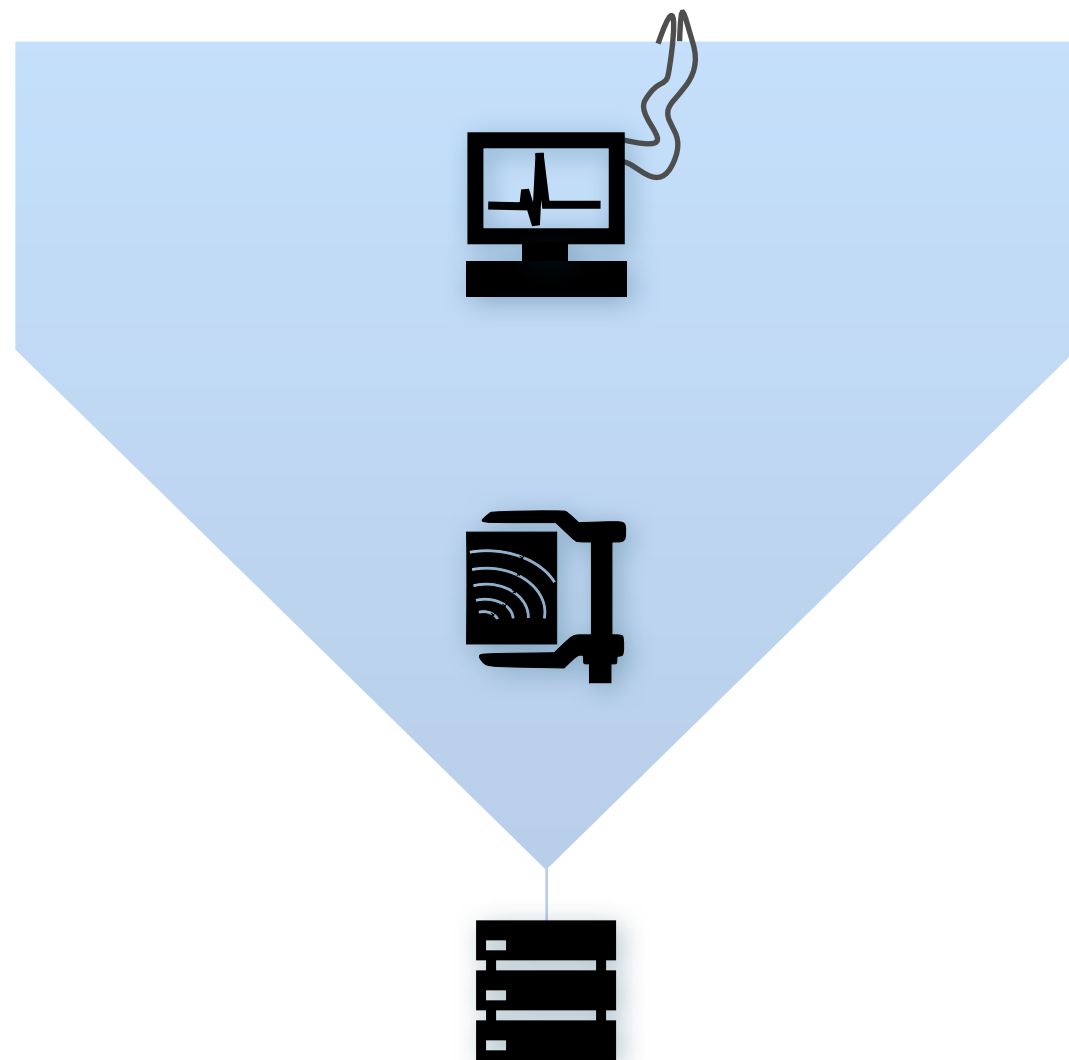
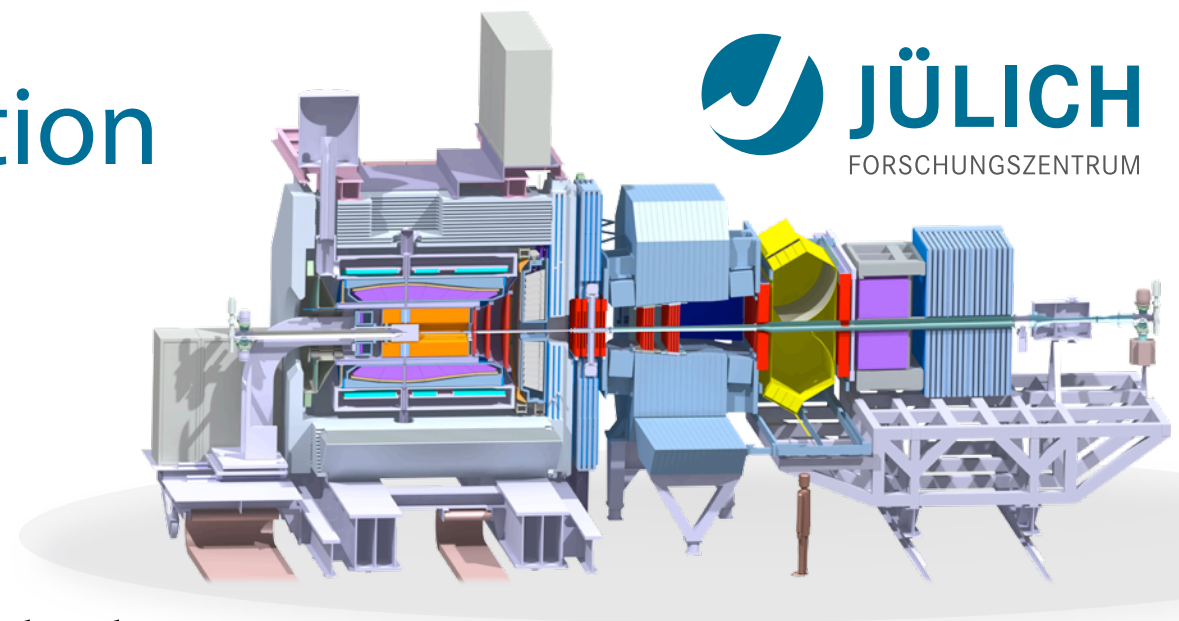
- Meson spectroscopy
 - Light mesons
 - Charmonium
 - Open charm
 - Exotic states
 - Glueballs
 - Hybrids
 - Molecules/multiquarks
- Baryon production
- Nucleon structure, e.m. processes
- Charm in nuclei
- Strangeness physics



→ Broad physics program

- **Continuous read out**

- *Novel feature*
- Background & signal similar
- No hardware trigger based on few sub-detectors, but online event reconstruction using full detector information



Rate

Event: $2 \times 10^7/\text{s}$

Raw data: 200 GB/s

Reduction

Amount: $\sim 1/1000$

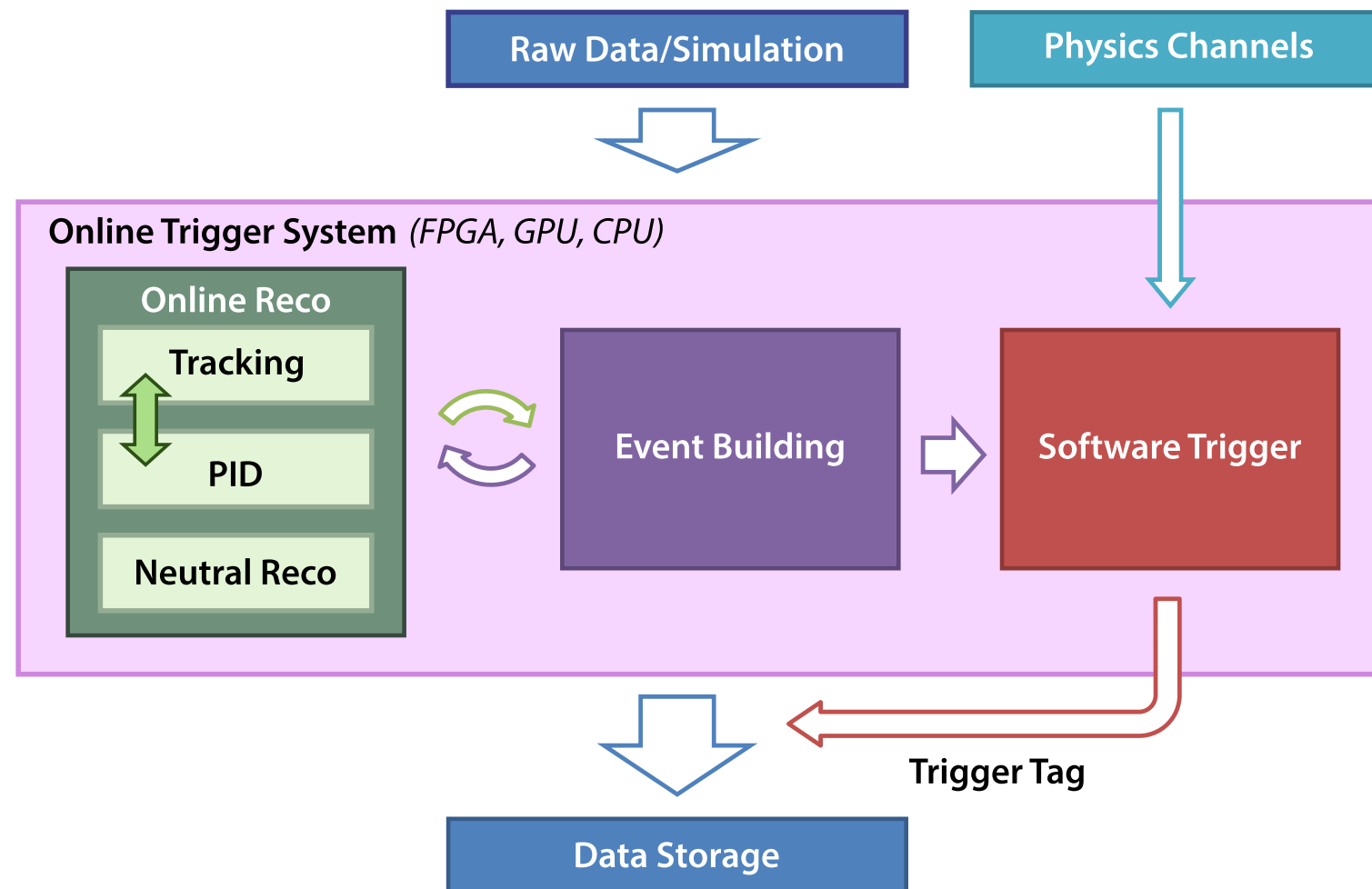
Time: 50 ns/evt

(Reject background events, save interesting events)

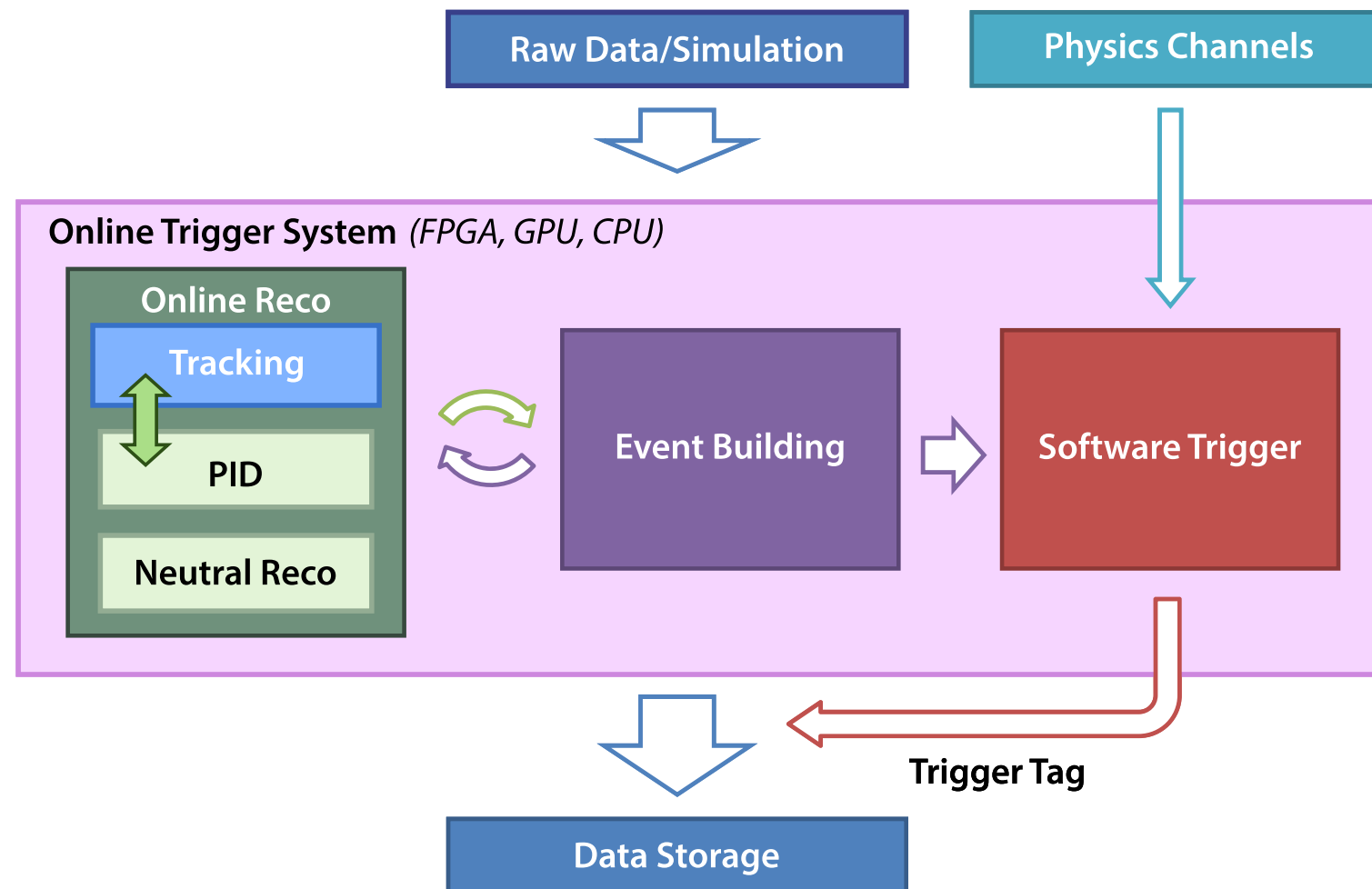
Storage space for
offline analysis

3 PB/y

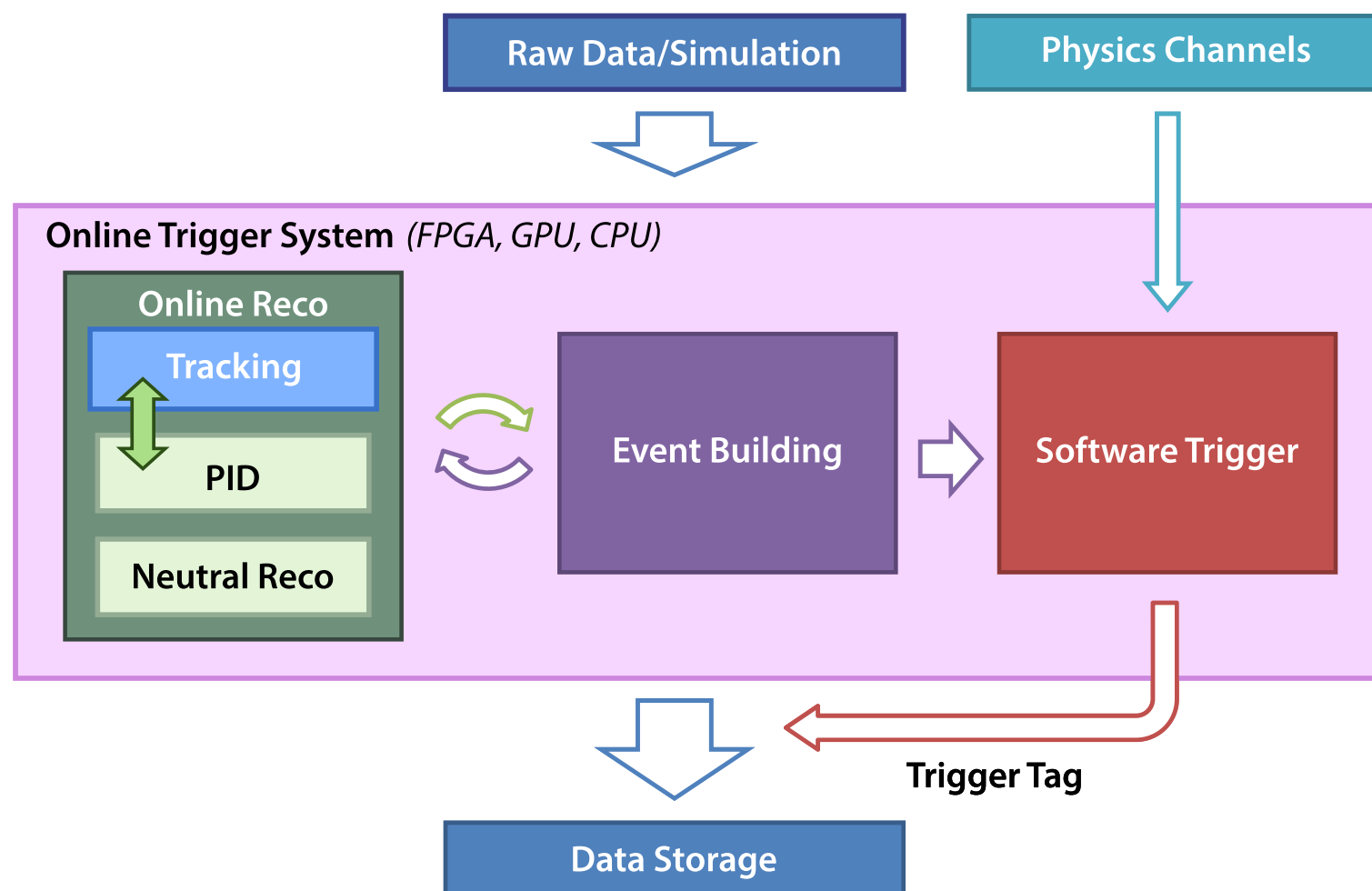
PANDA — Read Out Scheme



PANDA — Read Out Scheme



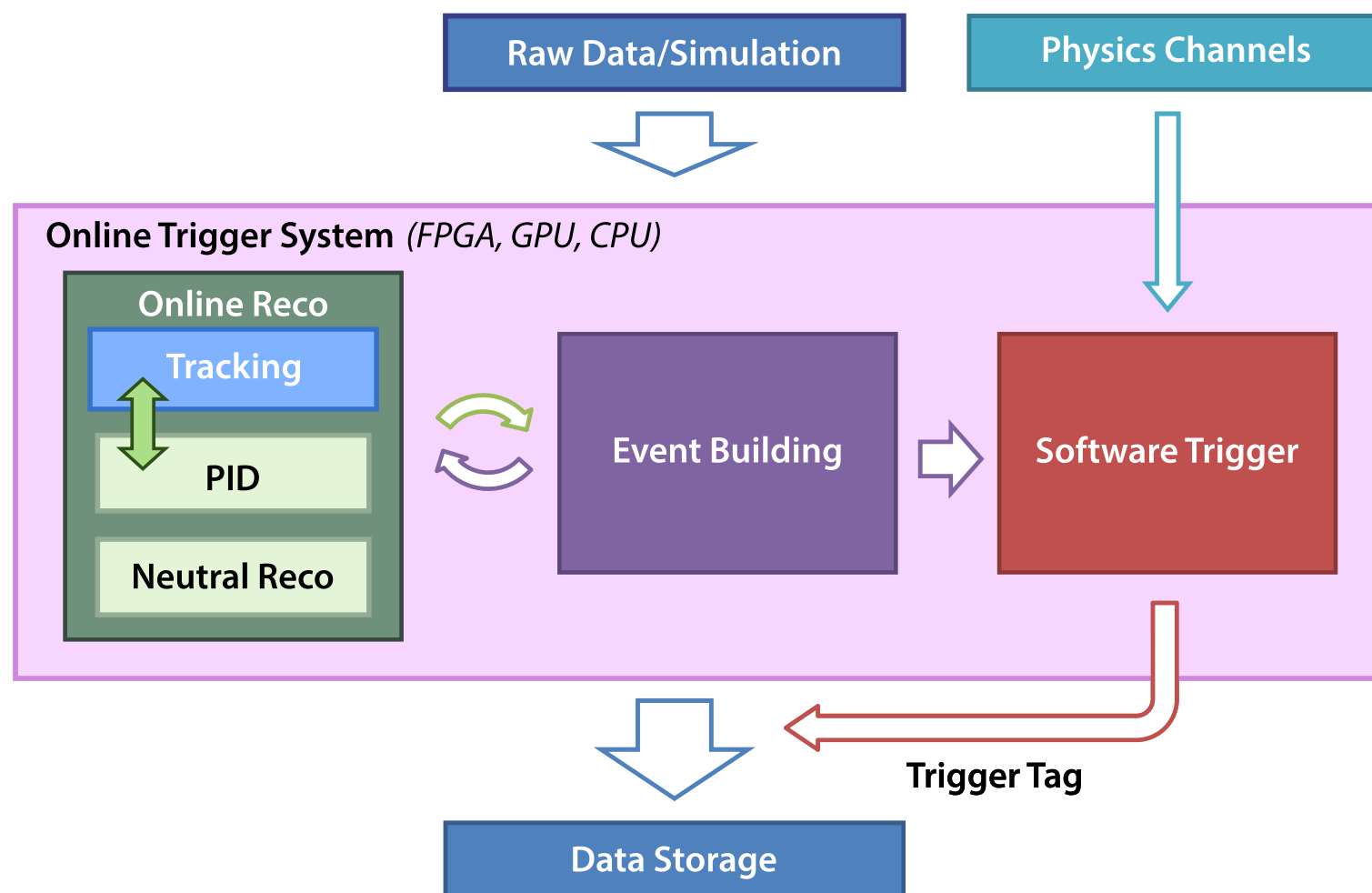
PANDA — Read Out Scheme



Requirements to Online Tracking

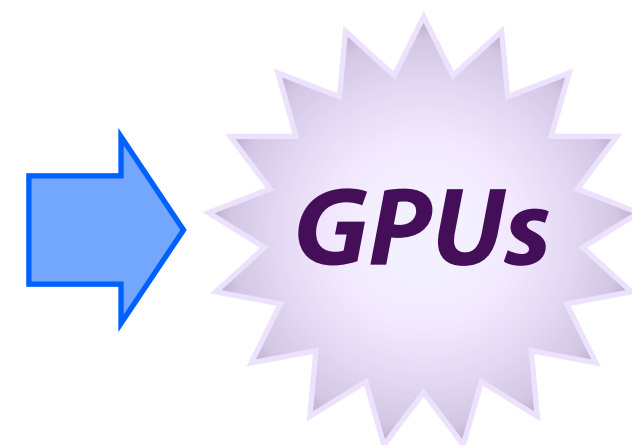
- Fast
- Sophisticated algorithms possible; reprogrammable
- Parallelism beyond single devices
- Fast
- *Limited precision ok*

PANDA — Read Out Scheme



Requirements to Online Tracking

- Fast
- Sophisticated algorithms possible; reprogrammable
- Parallelism beyond single devices
- Fast
- *Limited precision ok*



$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$

$\searrow \rightarrow e^+e^-$

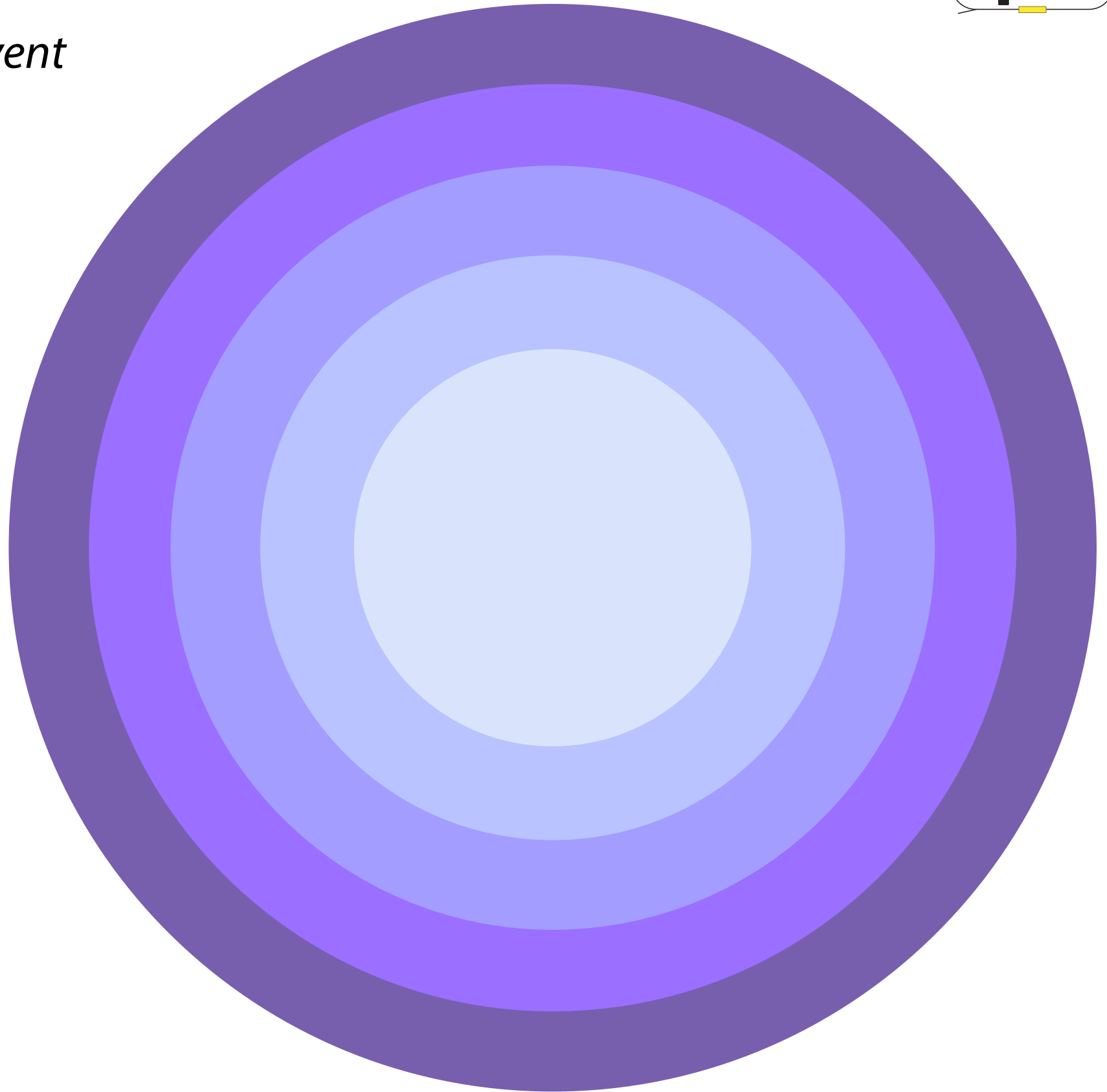
$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$

$\searrow \rightarrow e^+e^-$



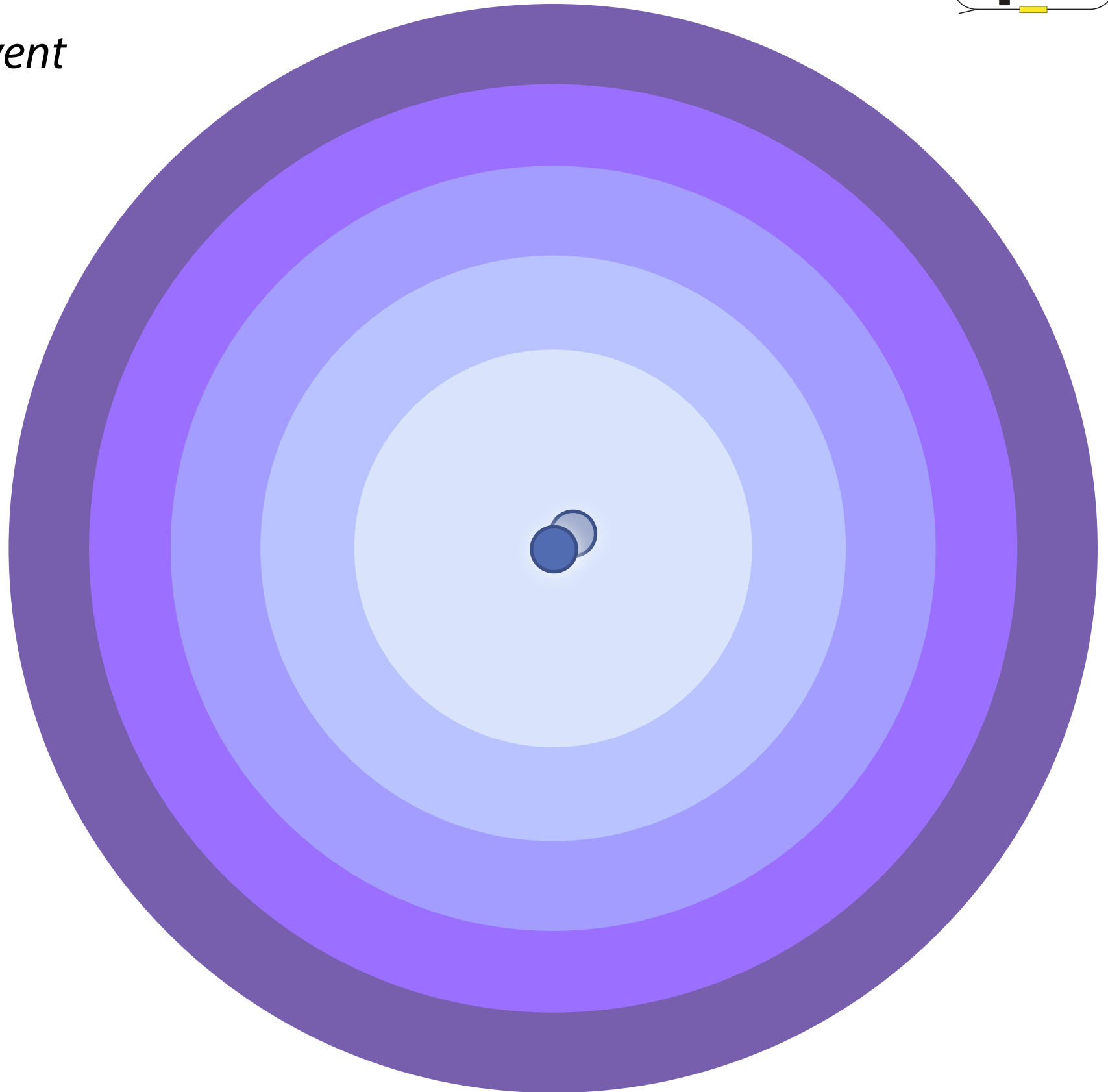
$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$

$\hookrightarrow e^+e^-$



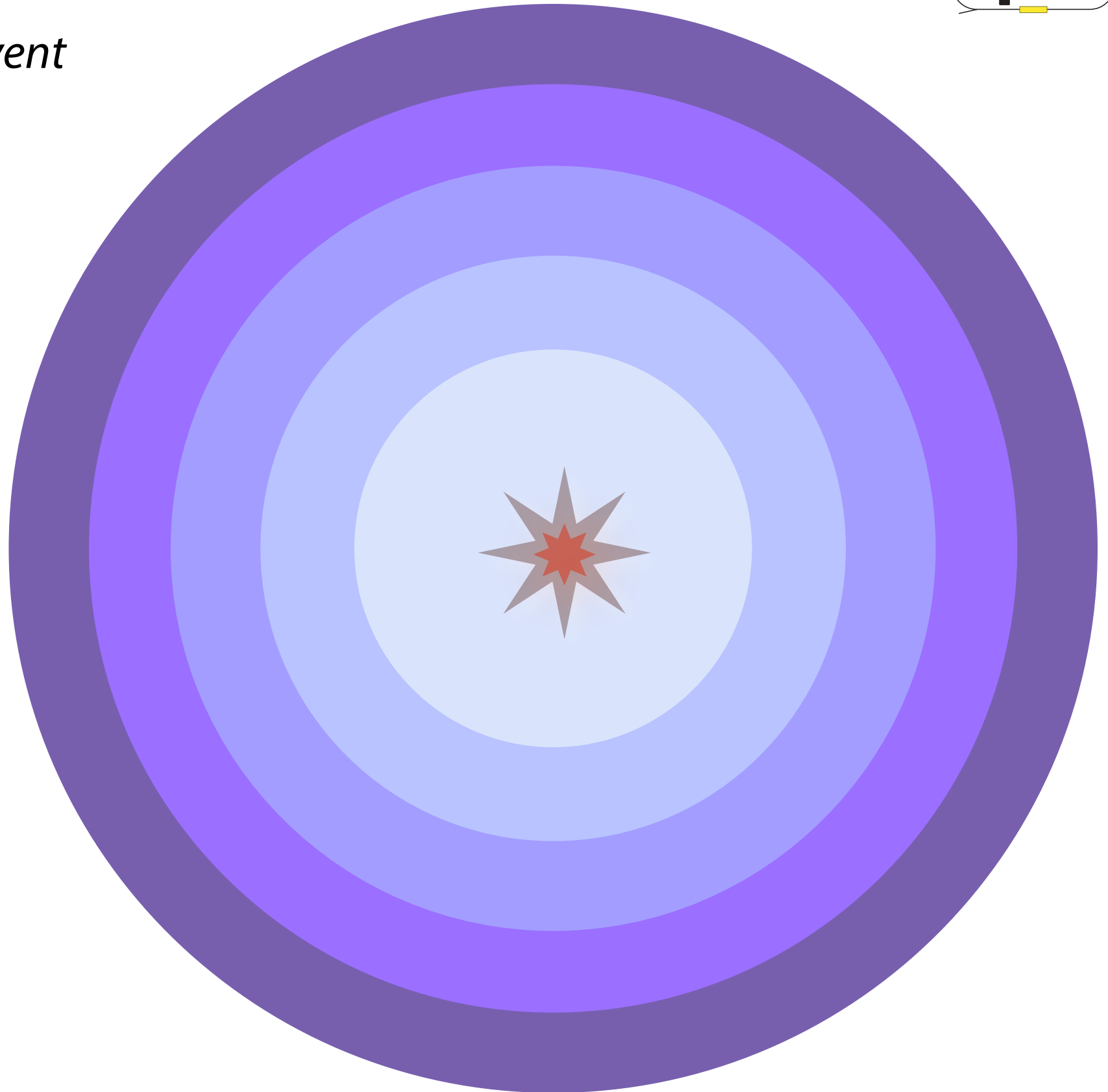
$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$

$\searrow \rightarrow e^+e^-$

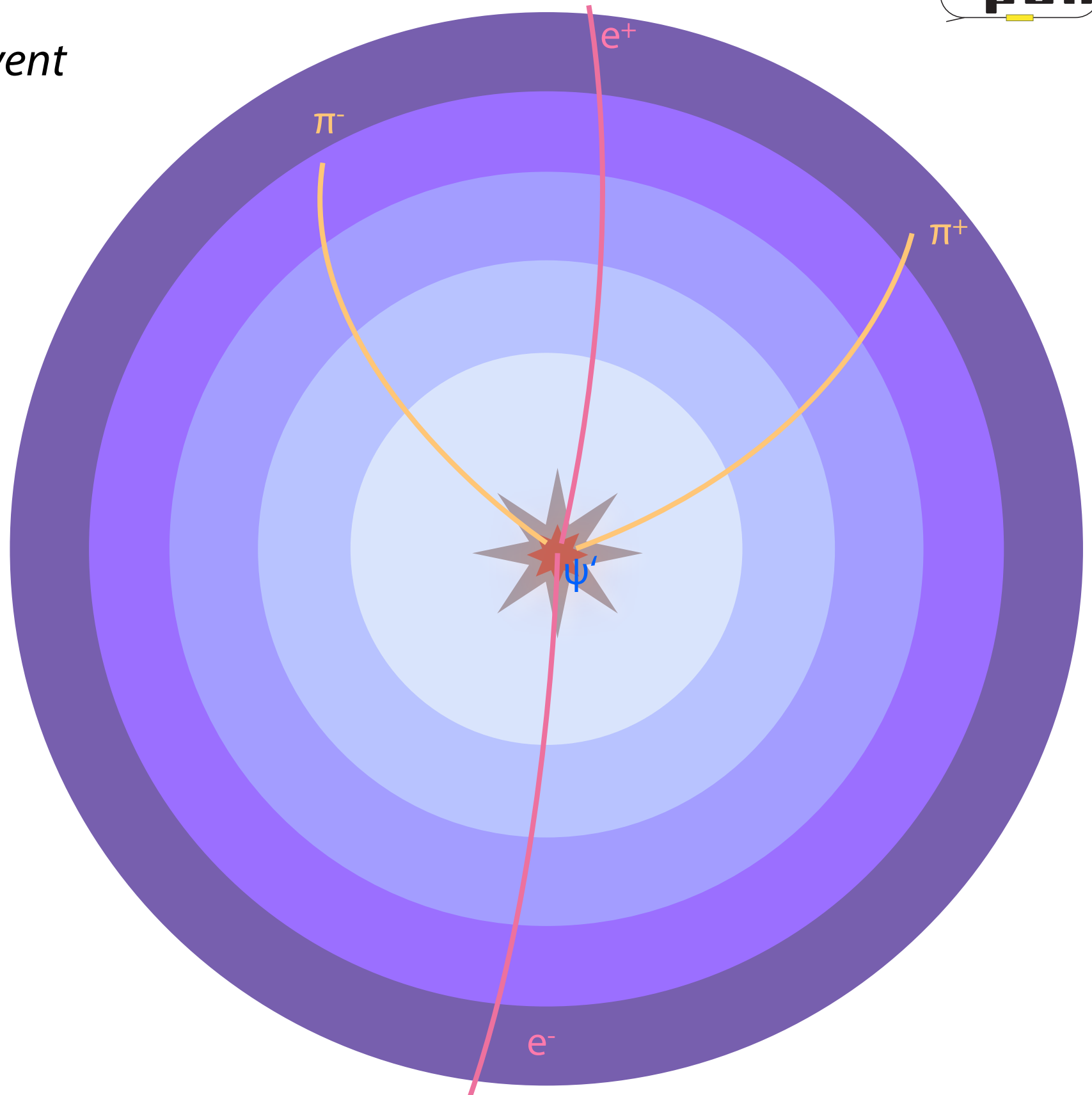


$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$
$$\quad \quad \quad \searrow \rightarrow e^+ e^-$$

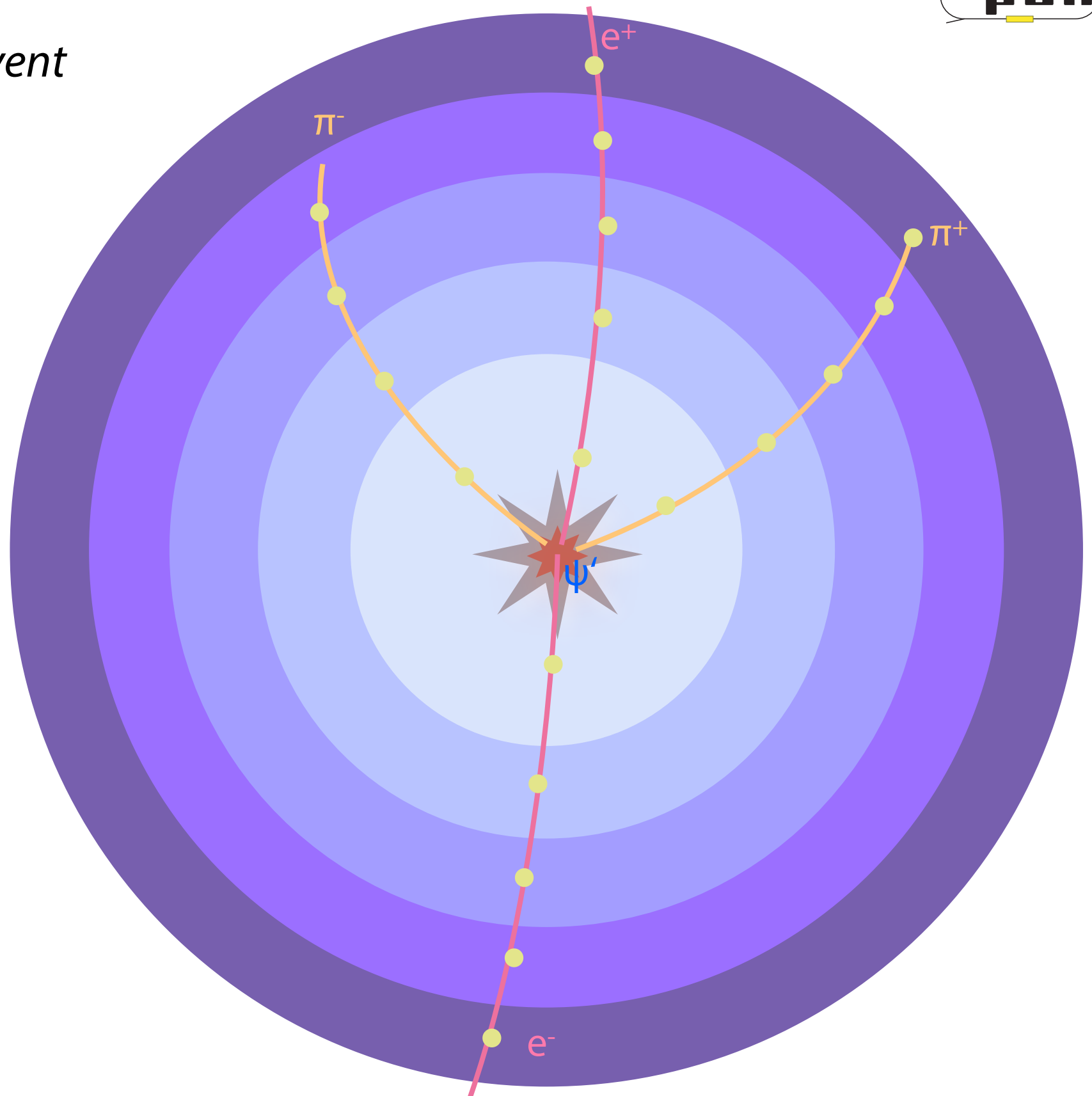


$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$
$$\quad \quad \quad \searrow \rightarrow e^+e^-$$

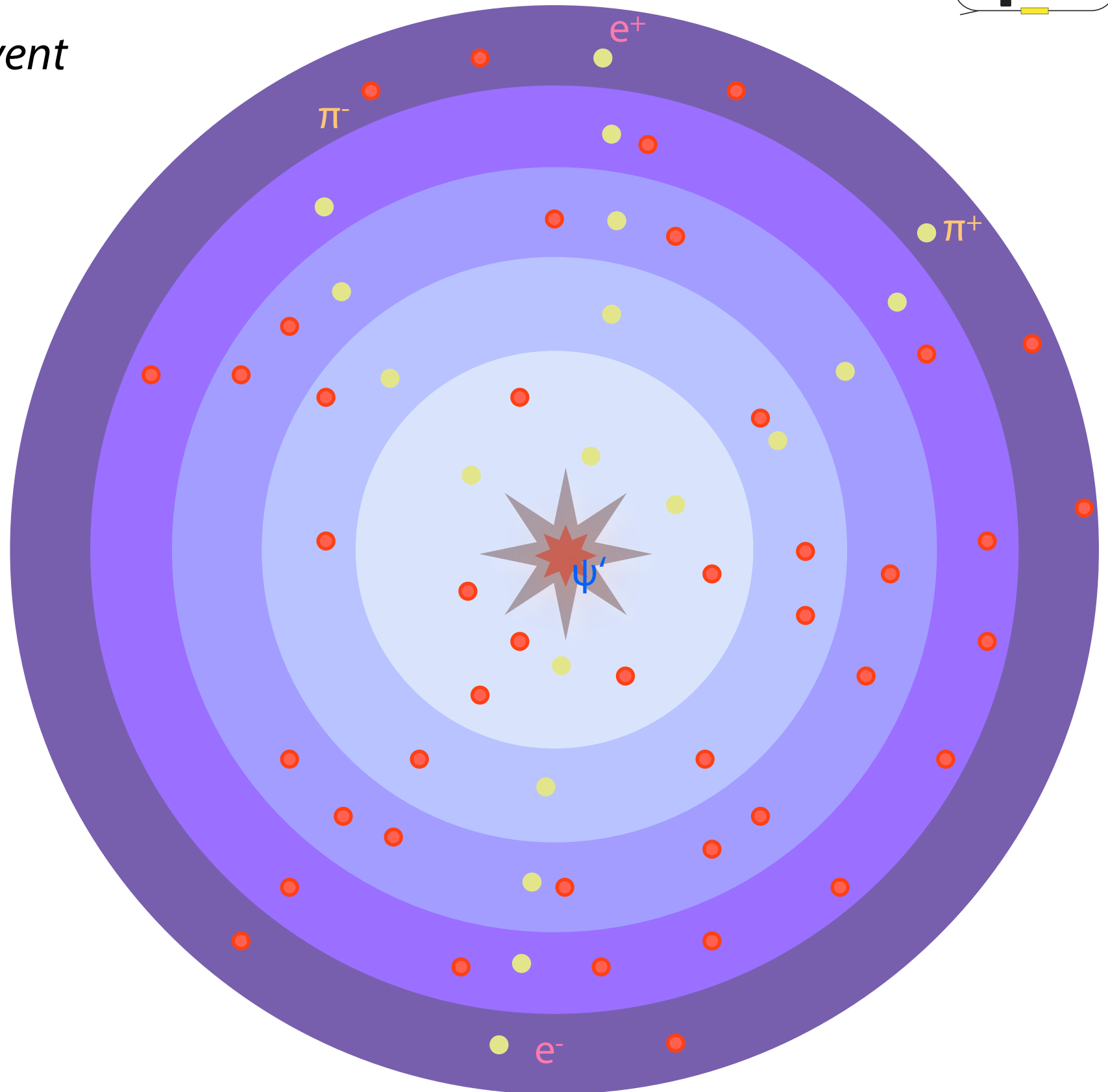


$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$
$$\quad \quad \quad \searrow \rightarrow e^+ e^-$$



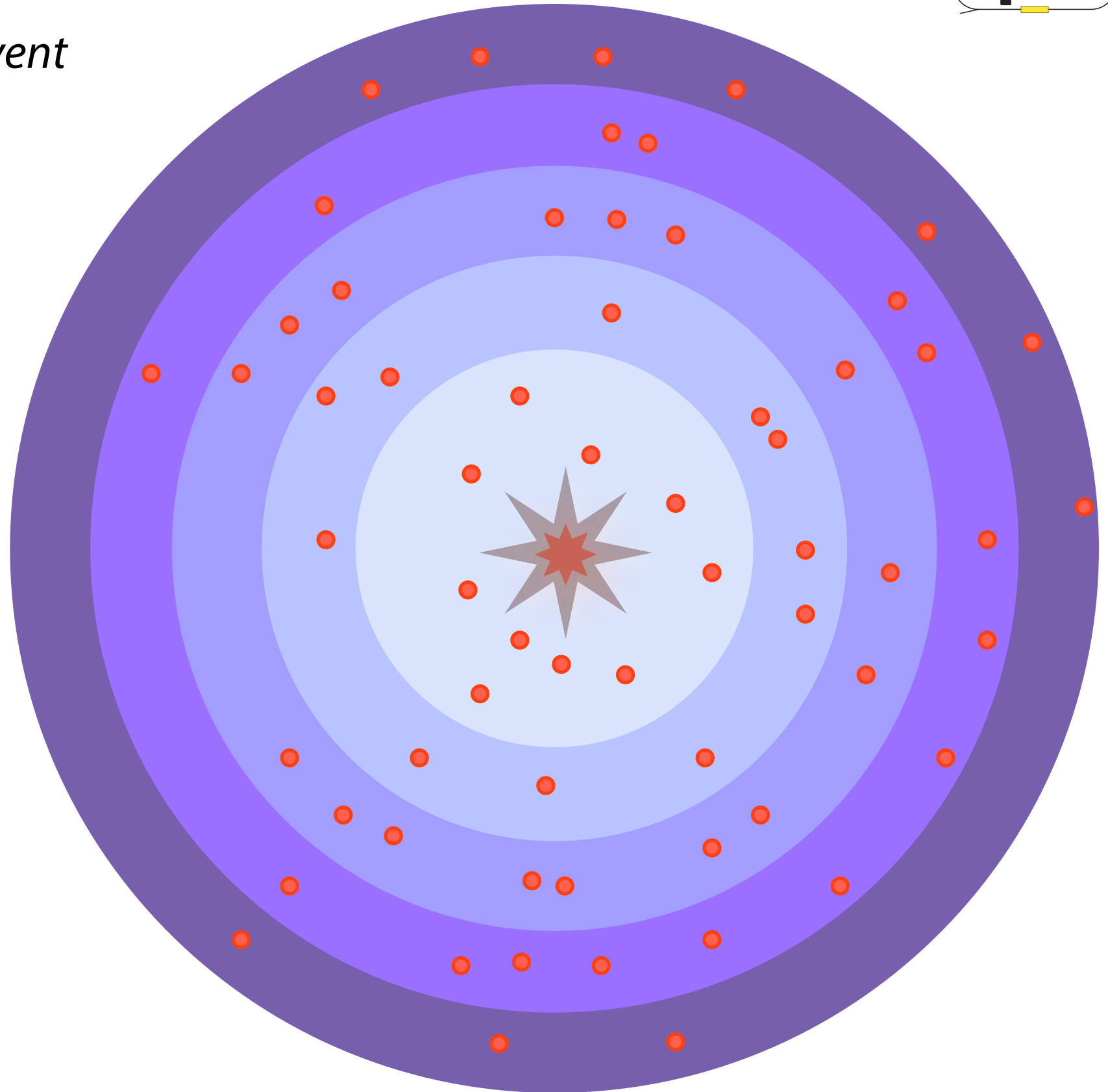
$\bar{P}$ ANDA — Online Tracking Example

The physics side:

Antiproton-proton event

$$p\bar{p} \rightarrow \psi' \rightarrow \psi \pi^+ \pi^-$$

$\hookrightarrow e^+e^-$



̄PANDA — Online Tracking Example

The detector side

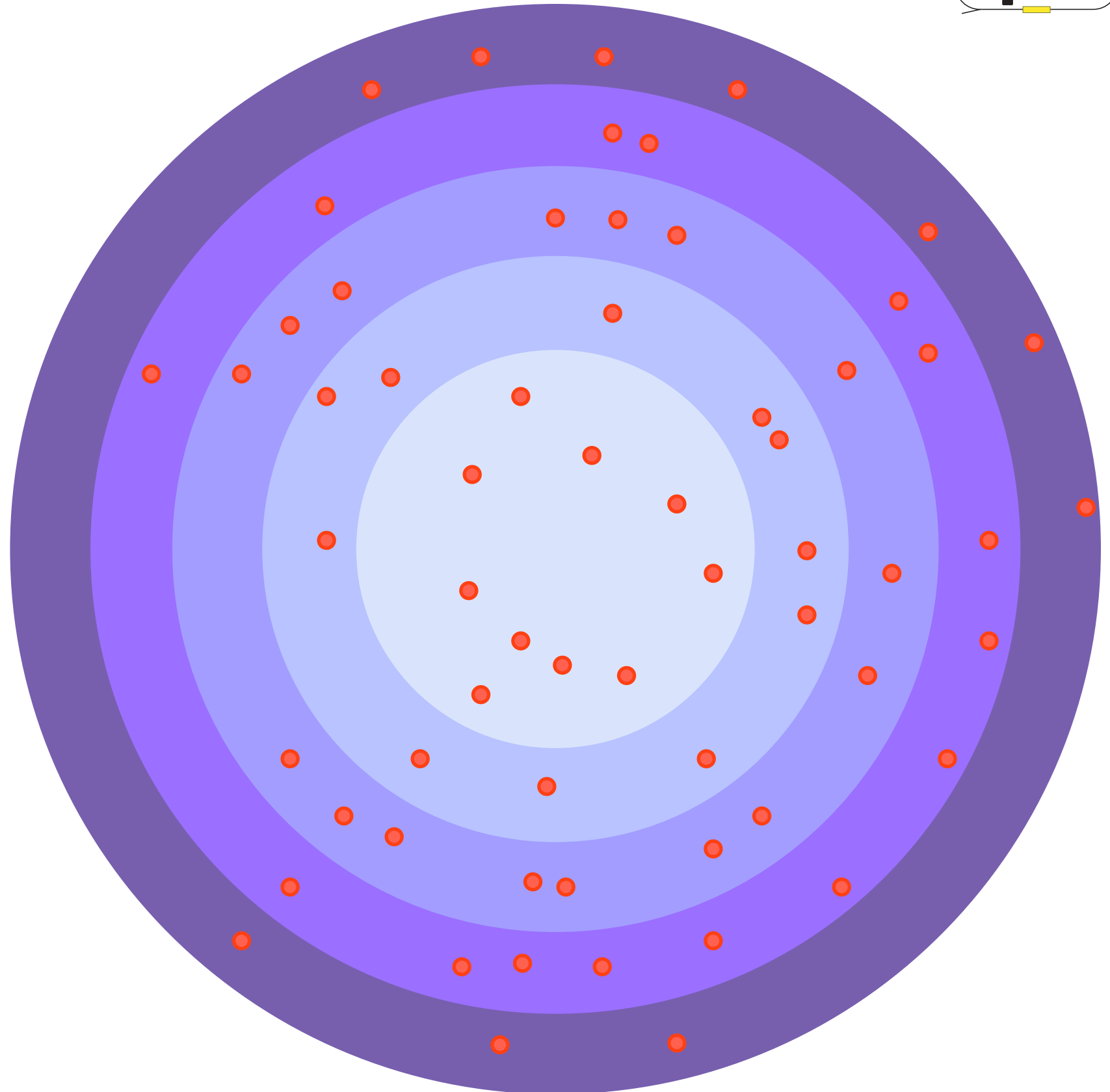
Everything in reverse



PANDA — Online Tracking Example

The detector side

Everything in reverse

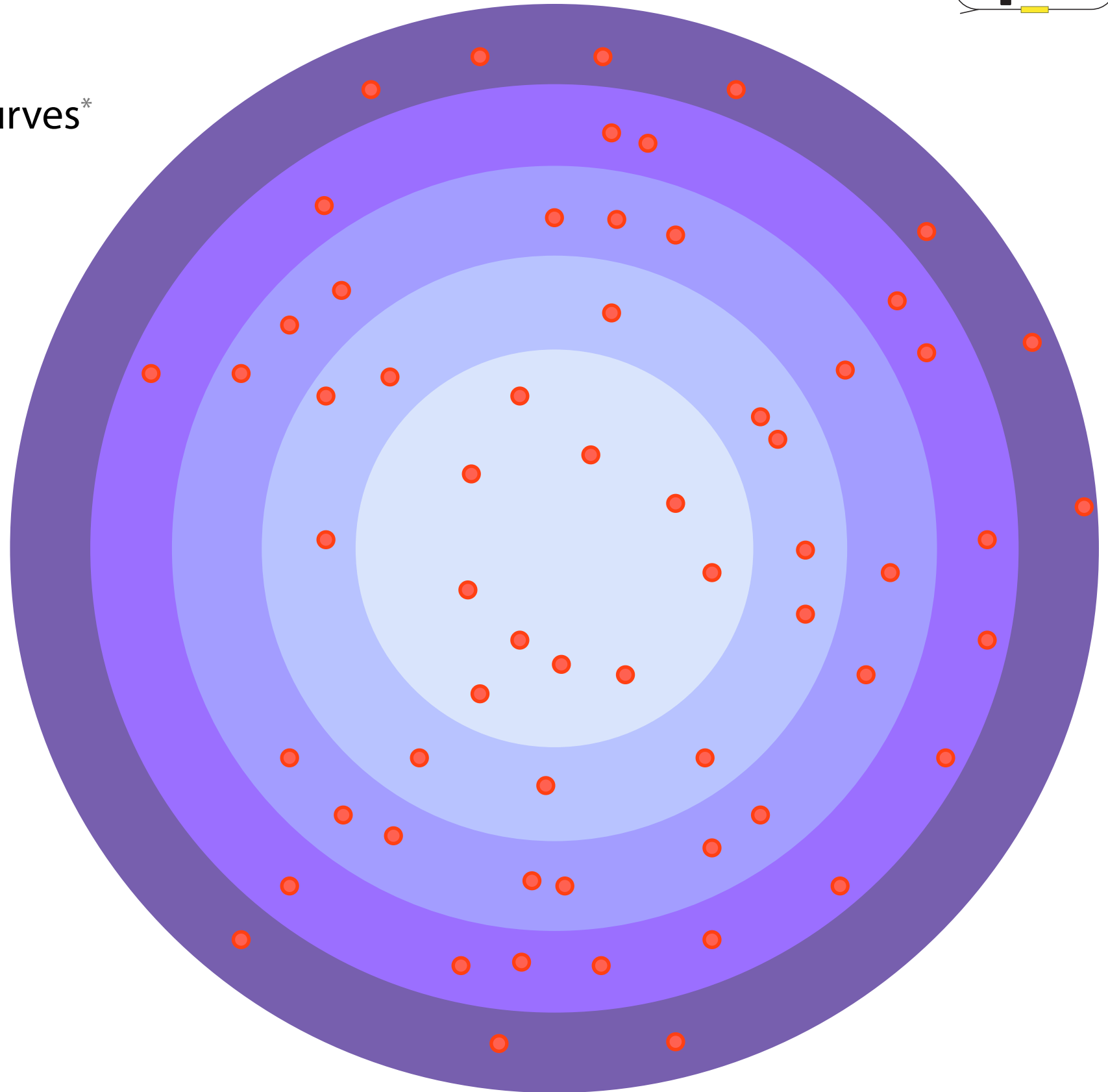


PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

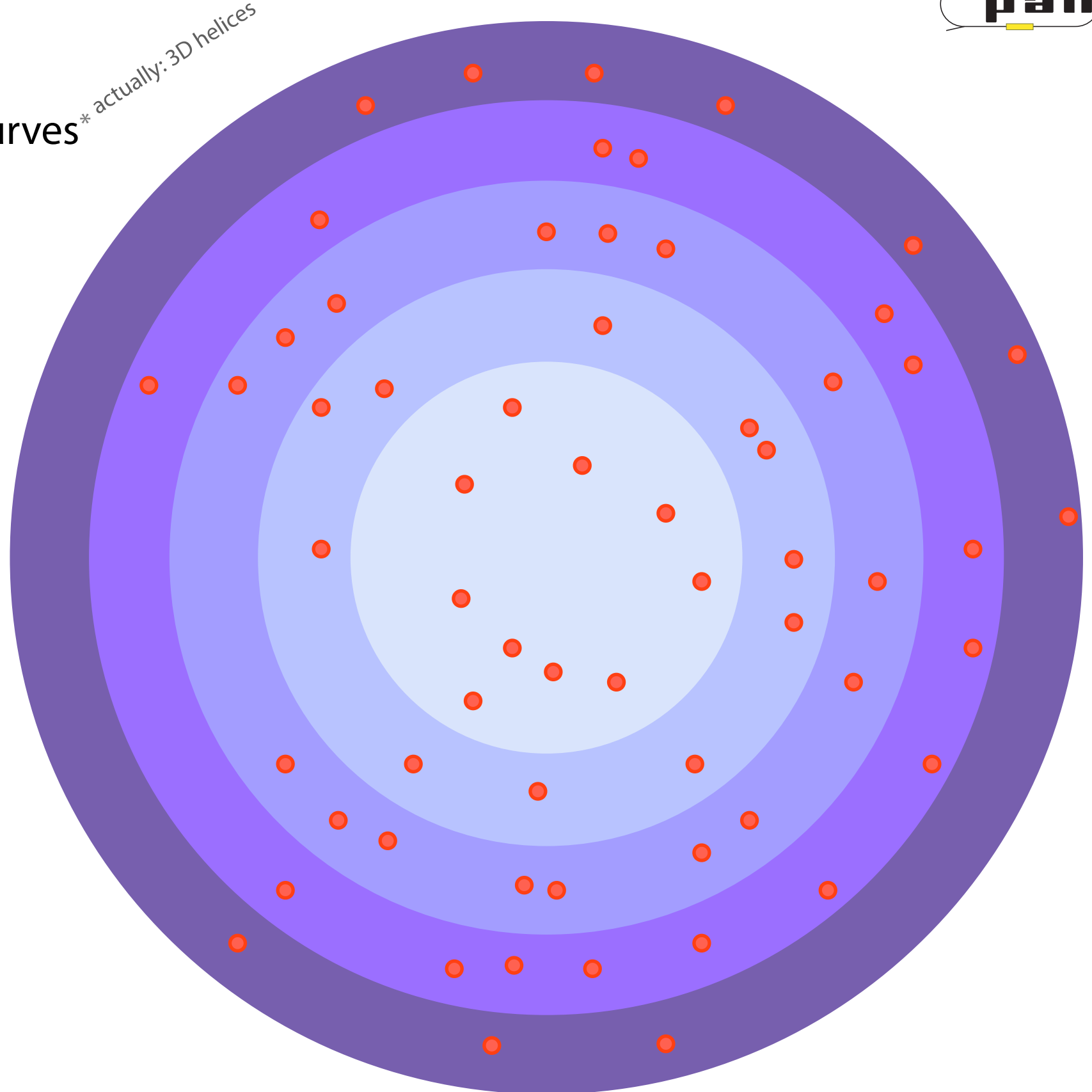


PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves* actually: 3D helices



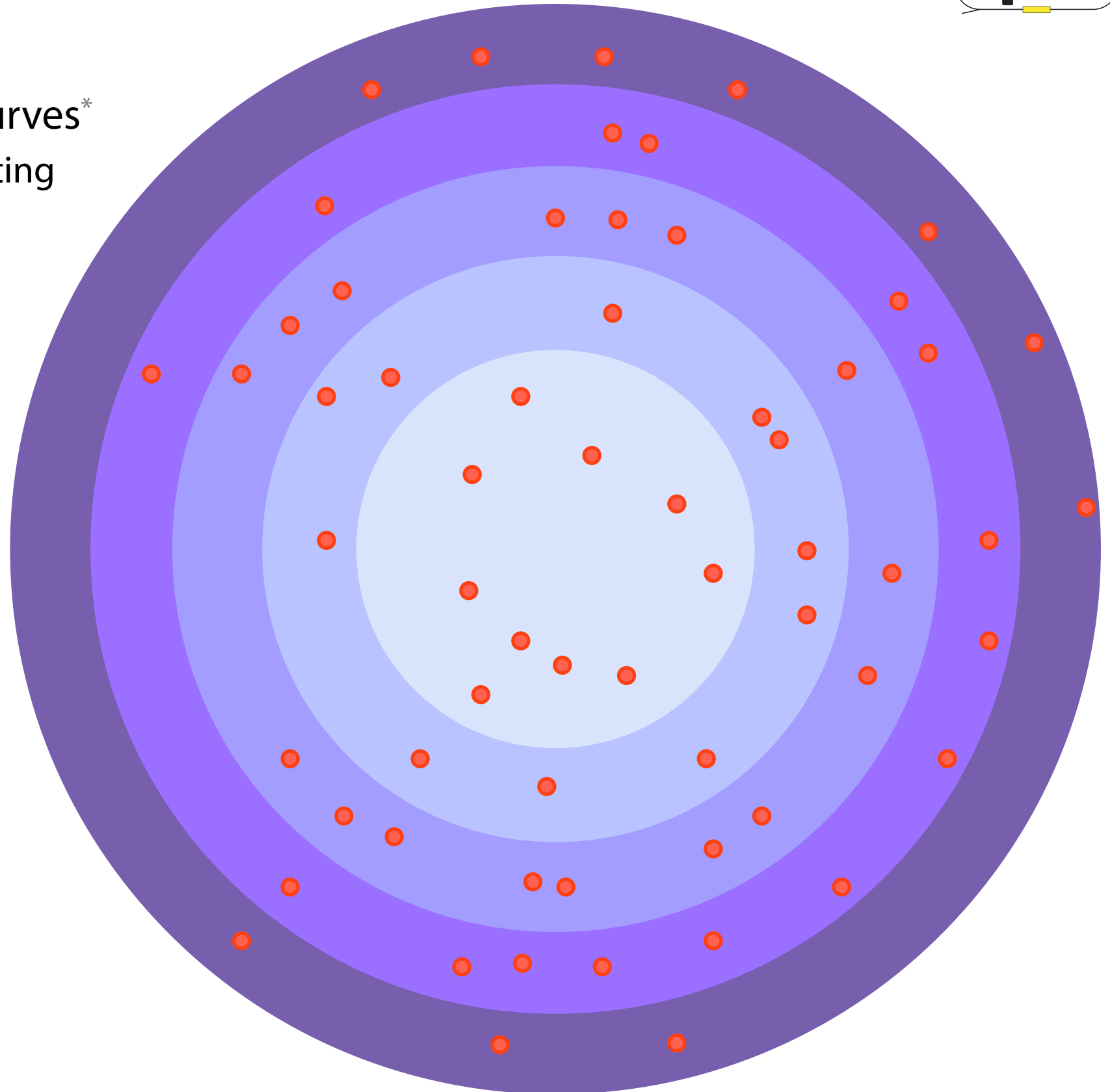
PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!



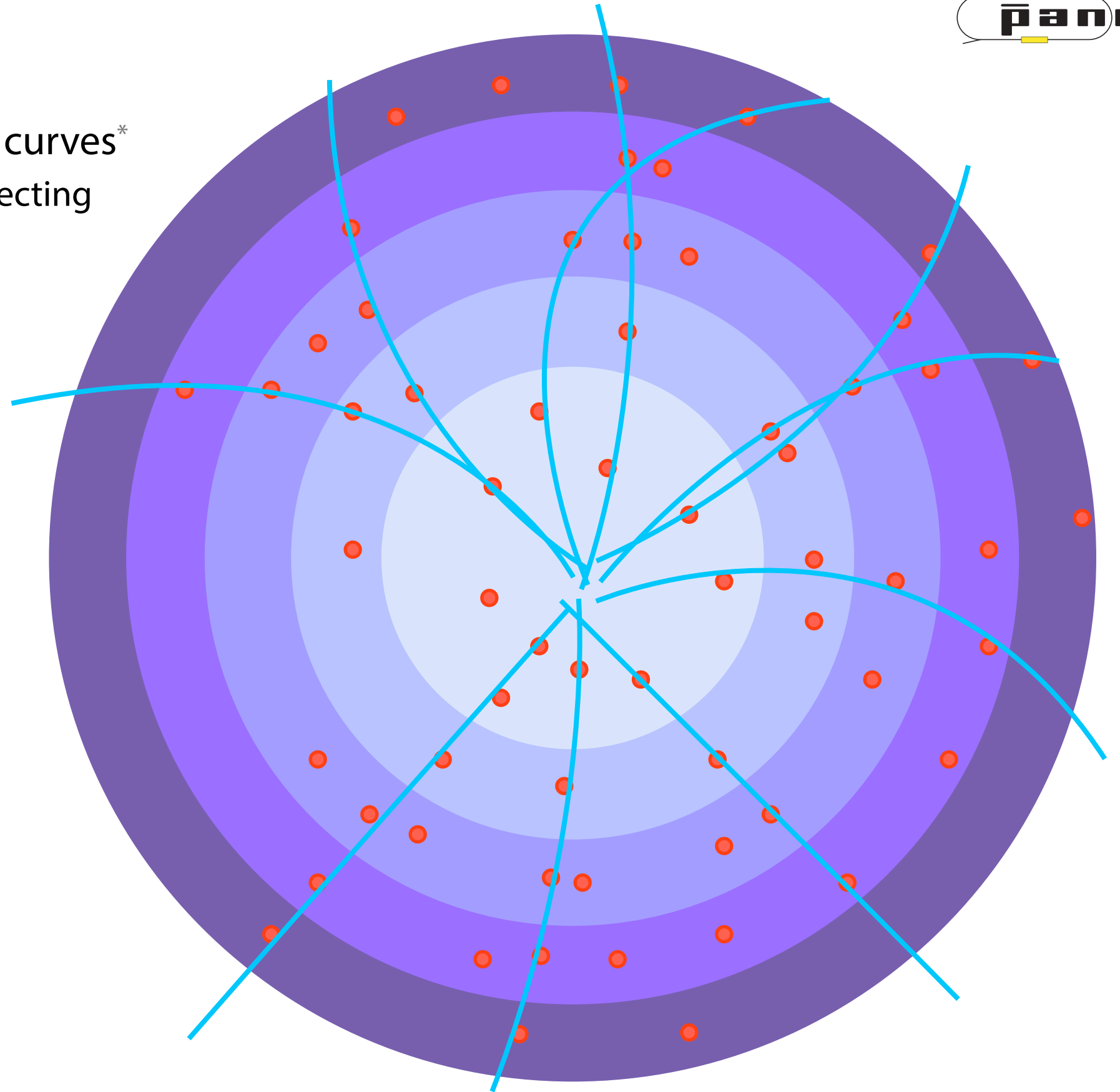
PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!



PANDA — Online Tracking Example

The detector side

Everything in reverse

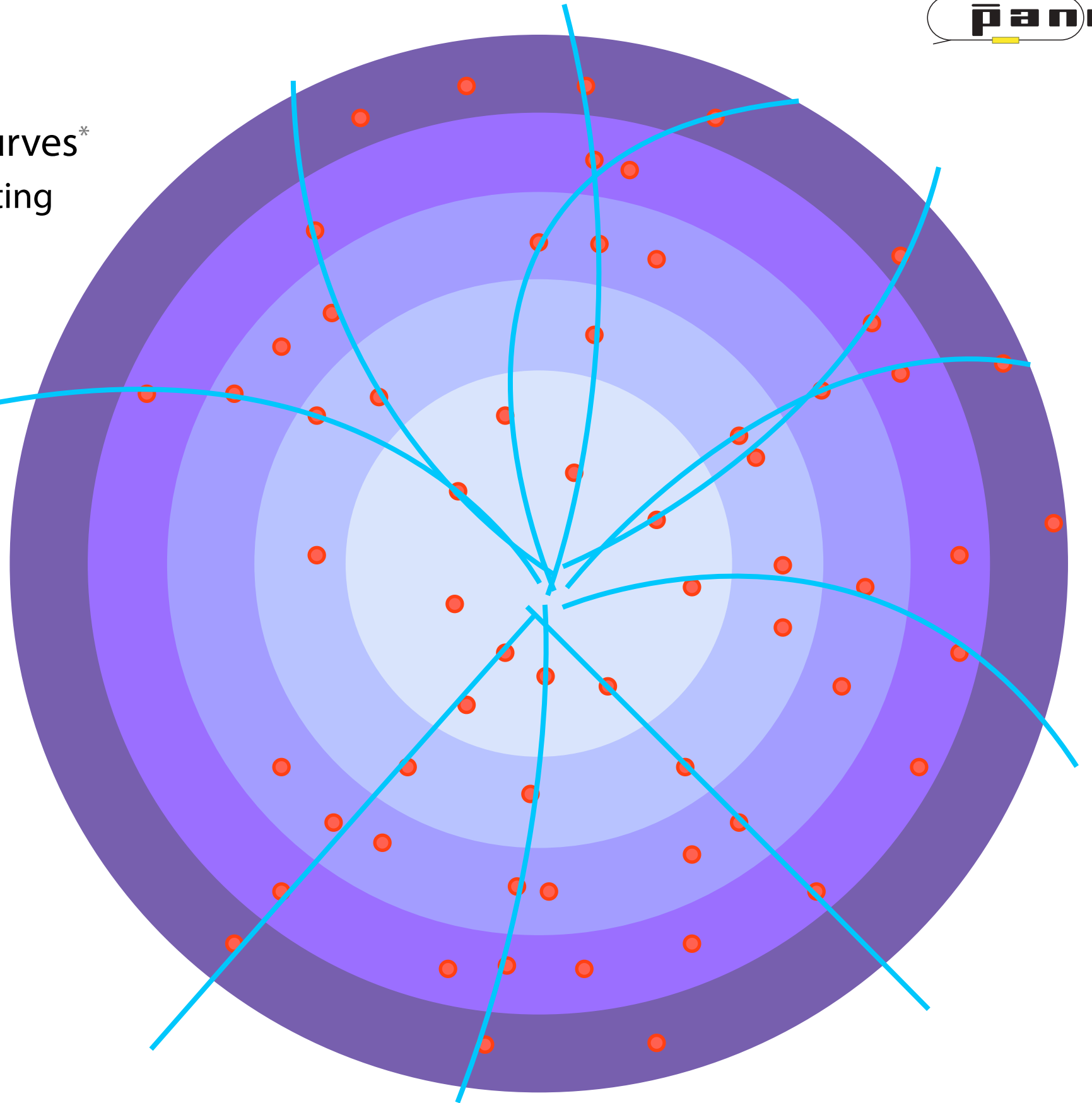
Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

How many hits?

Hits well matched?



PANDA — Online Tracking Example

The detector side

Everything in reverse

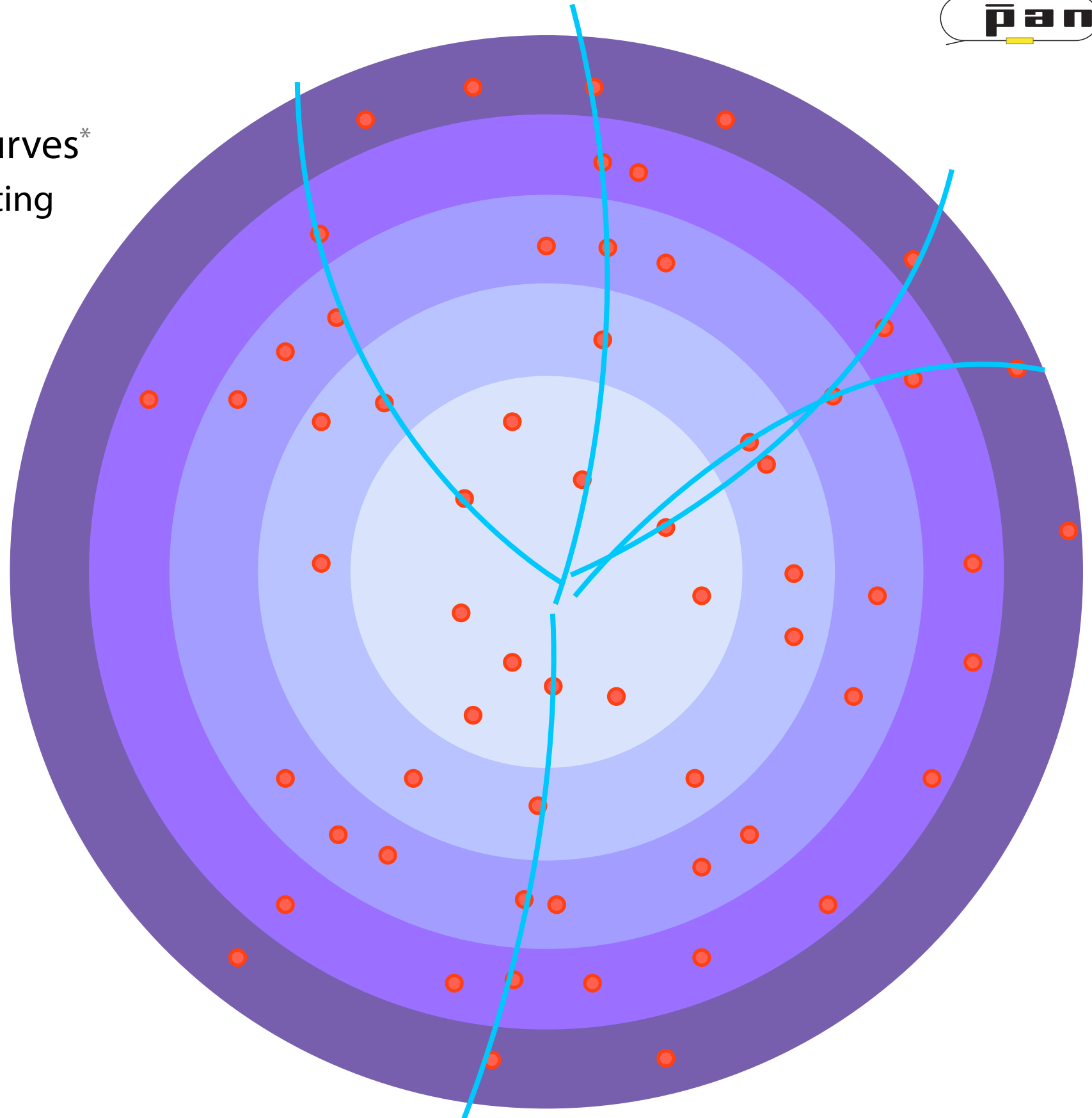
Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

How many hits?

Hits well matched?



PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

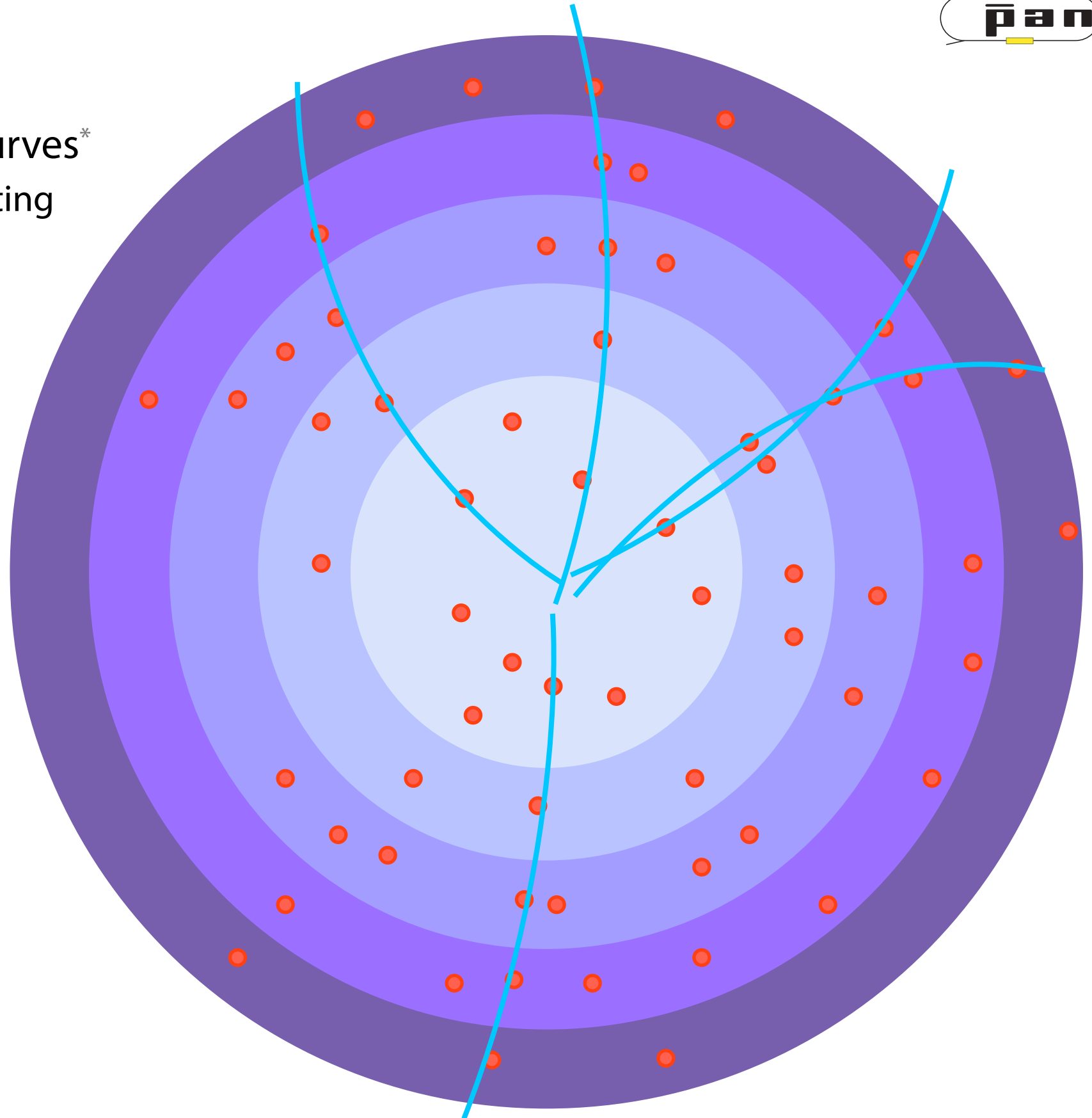
How many hits?

Hits well matched?

Identify final
particles

Curvature, length

...



PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

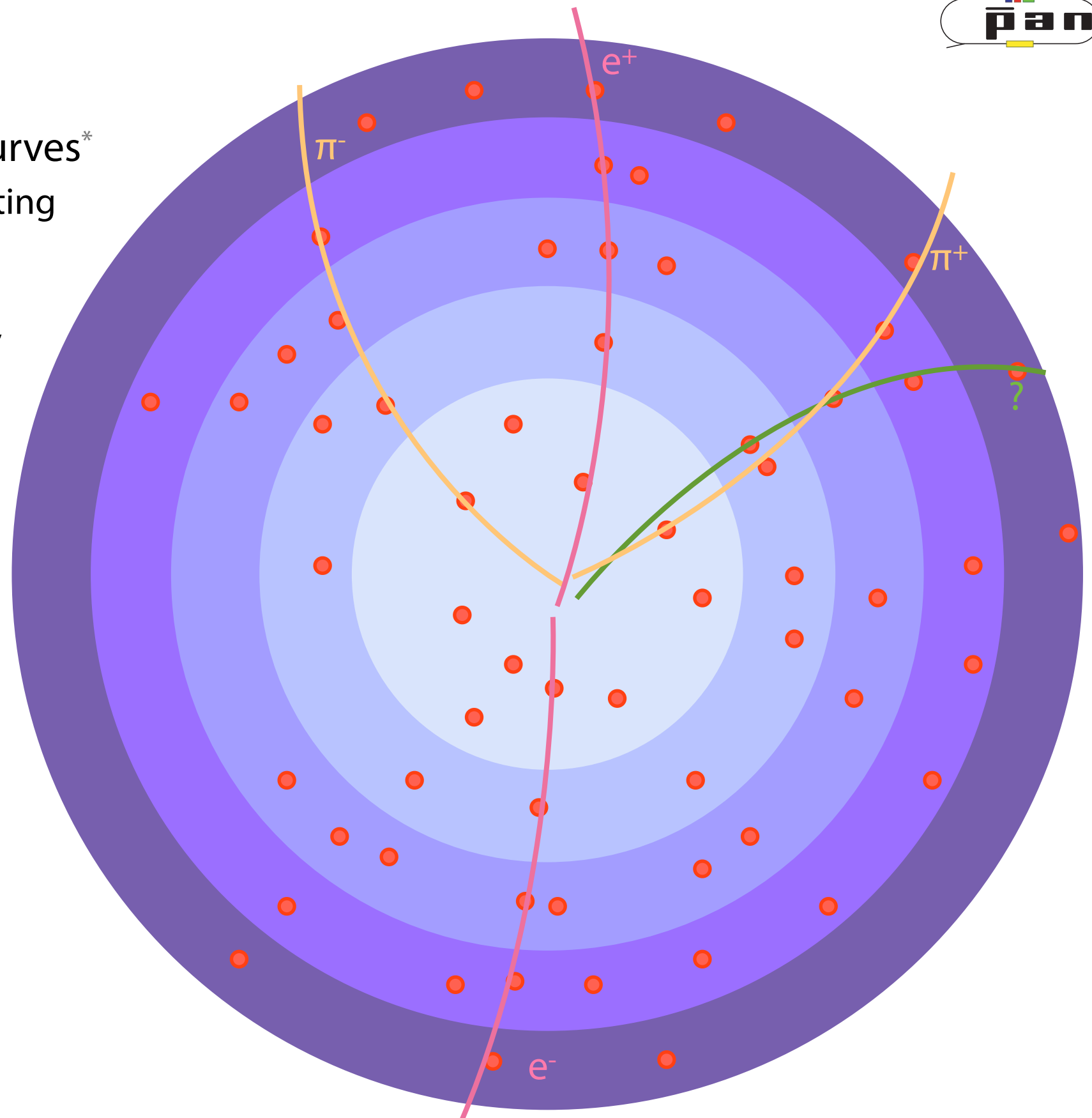
How many hits?

Hits well matched?

Identify final
particles

Curvature, length

...



PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

How many hits?

Hits well matched?

Identify final
particles

Curvature, length

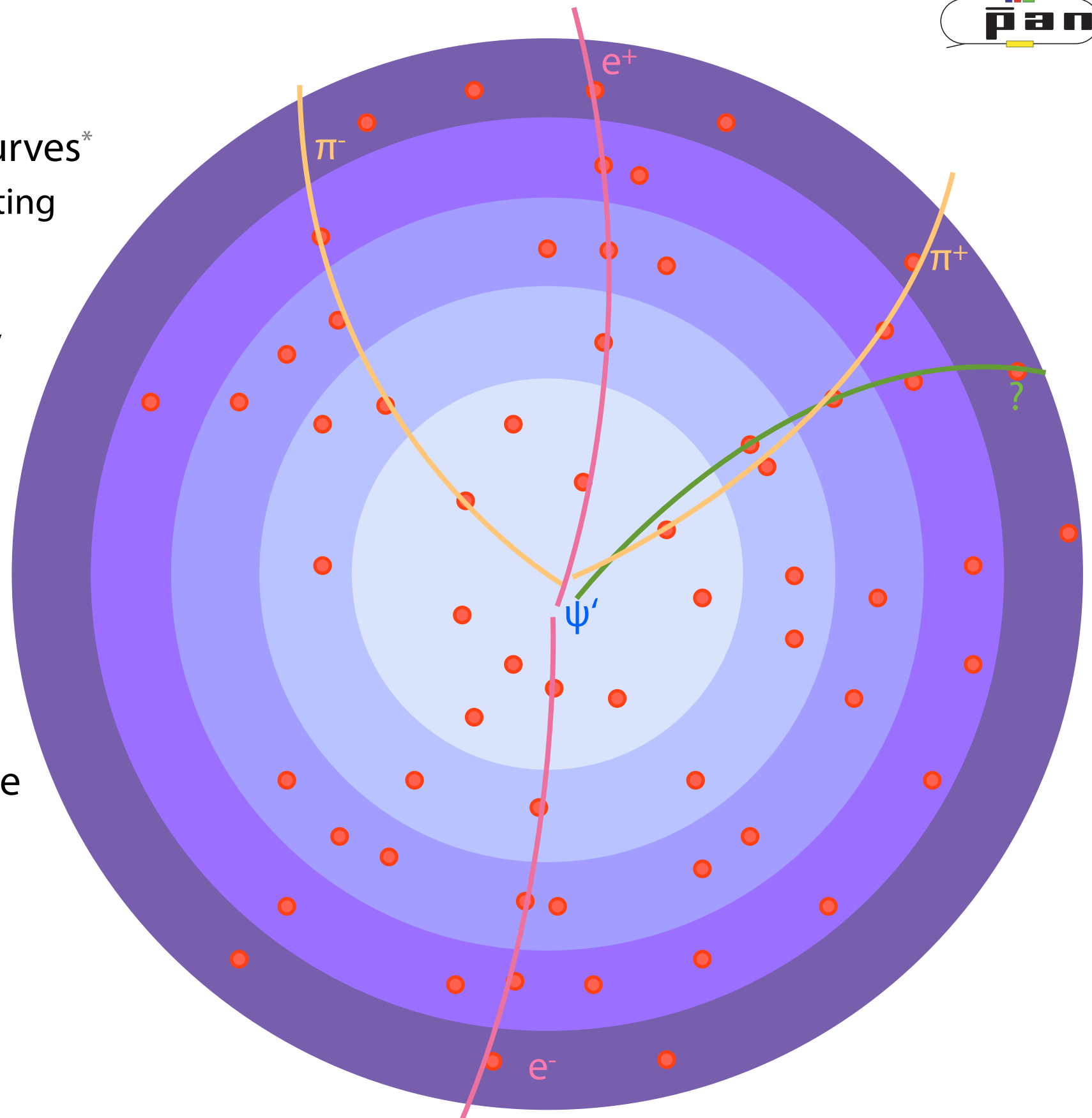
...

Identify intermediate
particles

Mass constraints

Geometry

...



PANDA — Online Tracking Example

The detector side

Everything in reverse

Particle tracks are curves*

→ Find curves connecting
hit points!

Sort by track quality

How many hits?

Hits well matched?

Identify final
particles

Curvature, length

...

Identify intermediate
particles

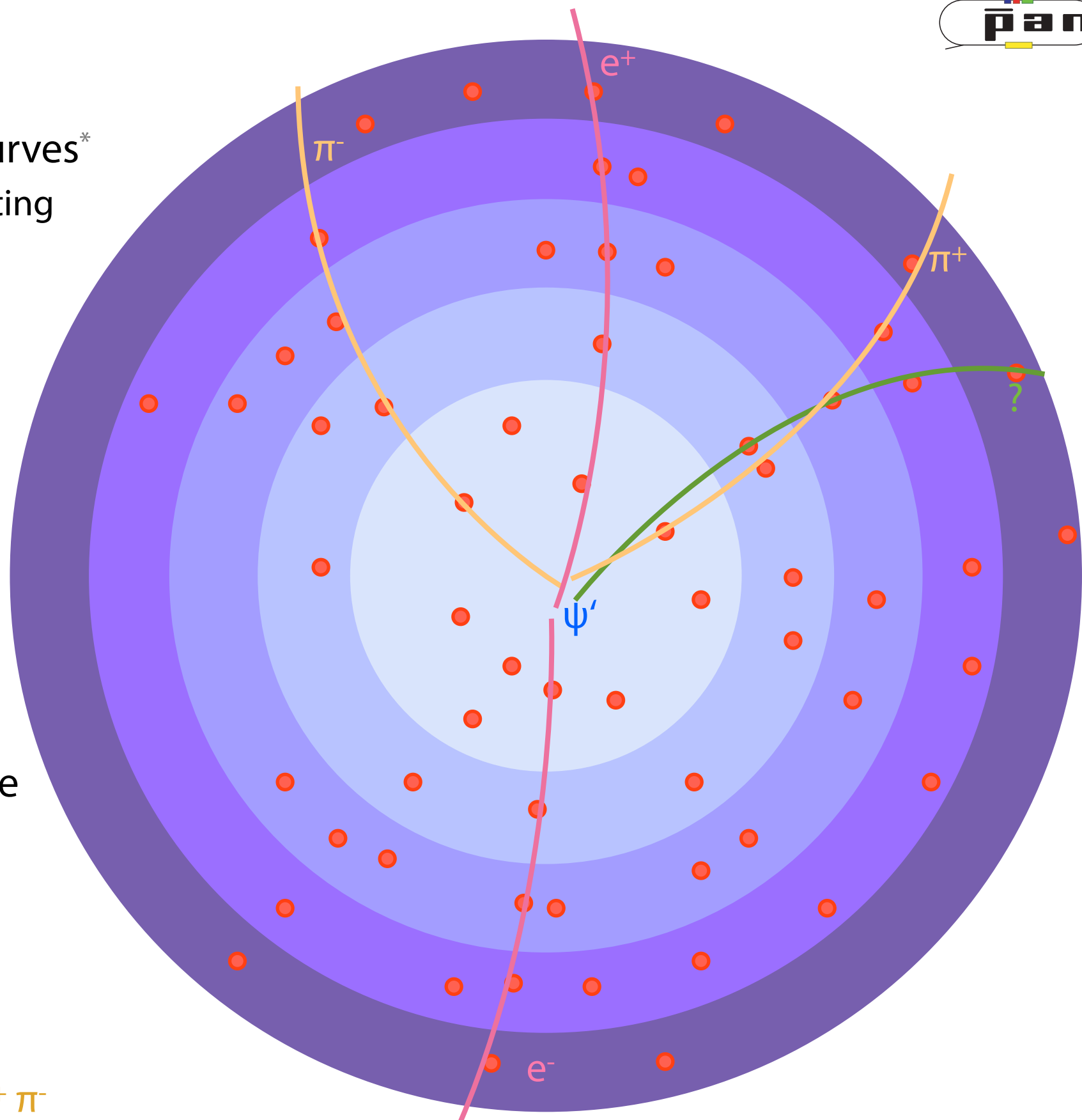
Mass constraints

Geometry

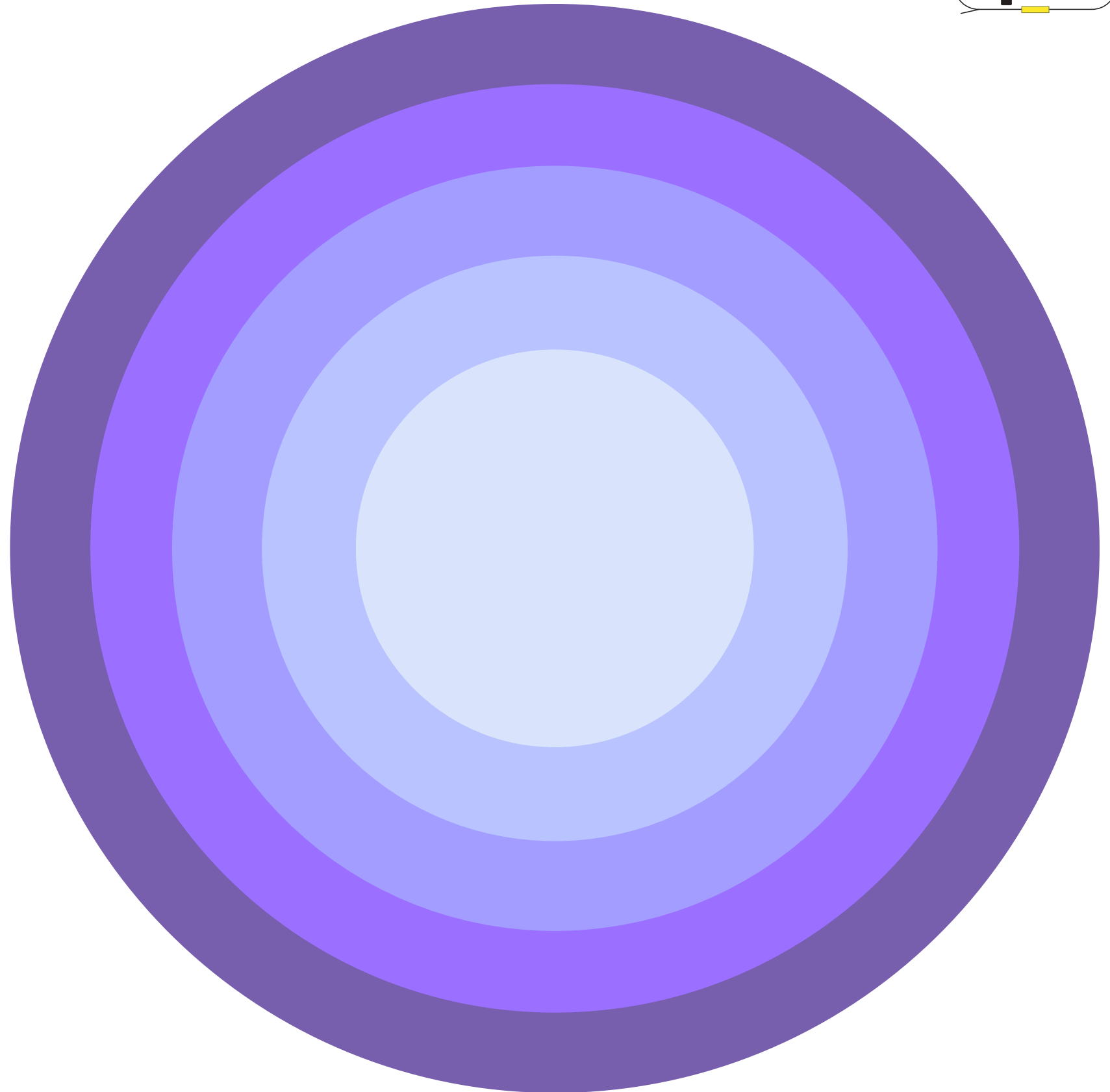
...

Identify process:

$p\bar{p} \rightarrow \psi' \rightarrow e^+e^- \pi^+ \pi^-$

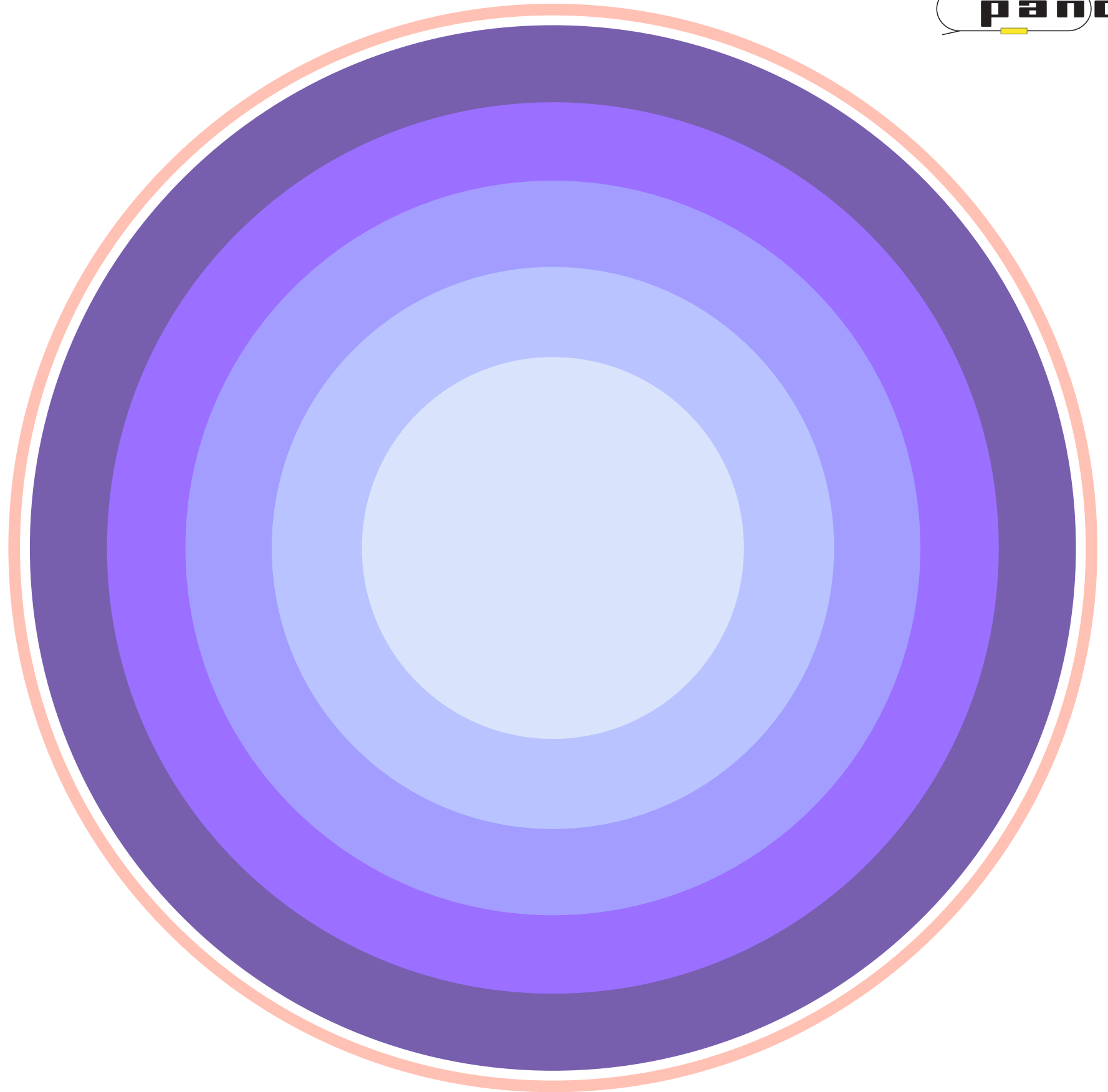


PANDA — Triggering



$\bar{\text{P}}$ ANDA — Triggering

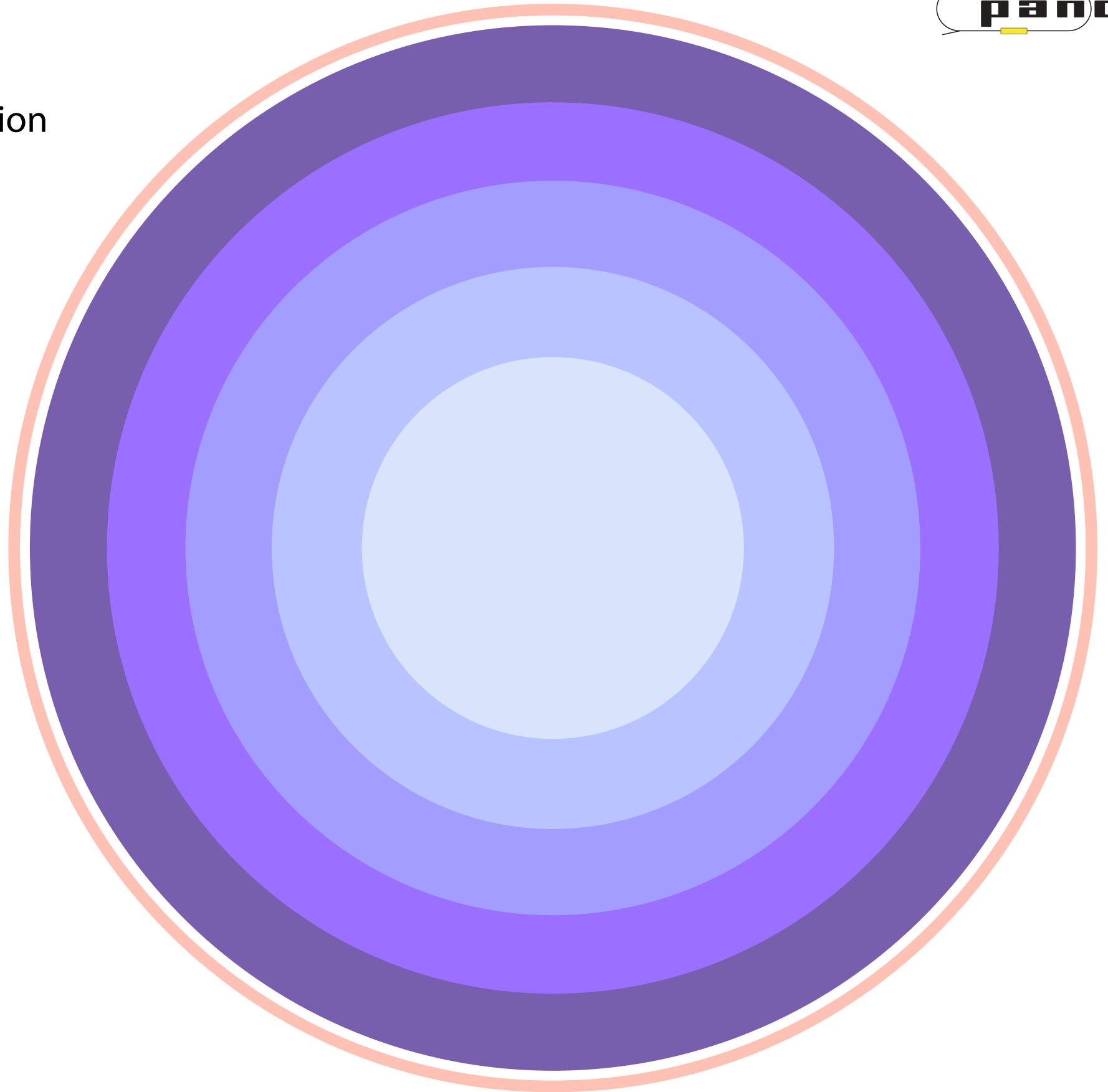
Trigger



PANDA — Triggering

Trigger

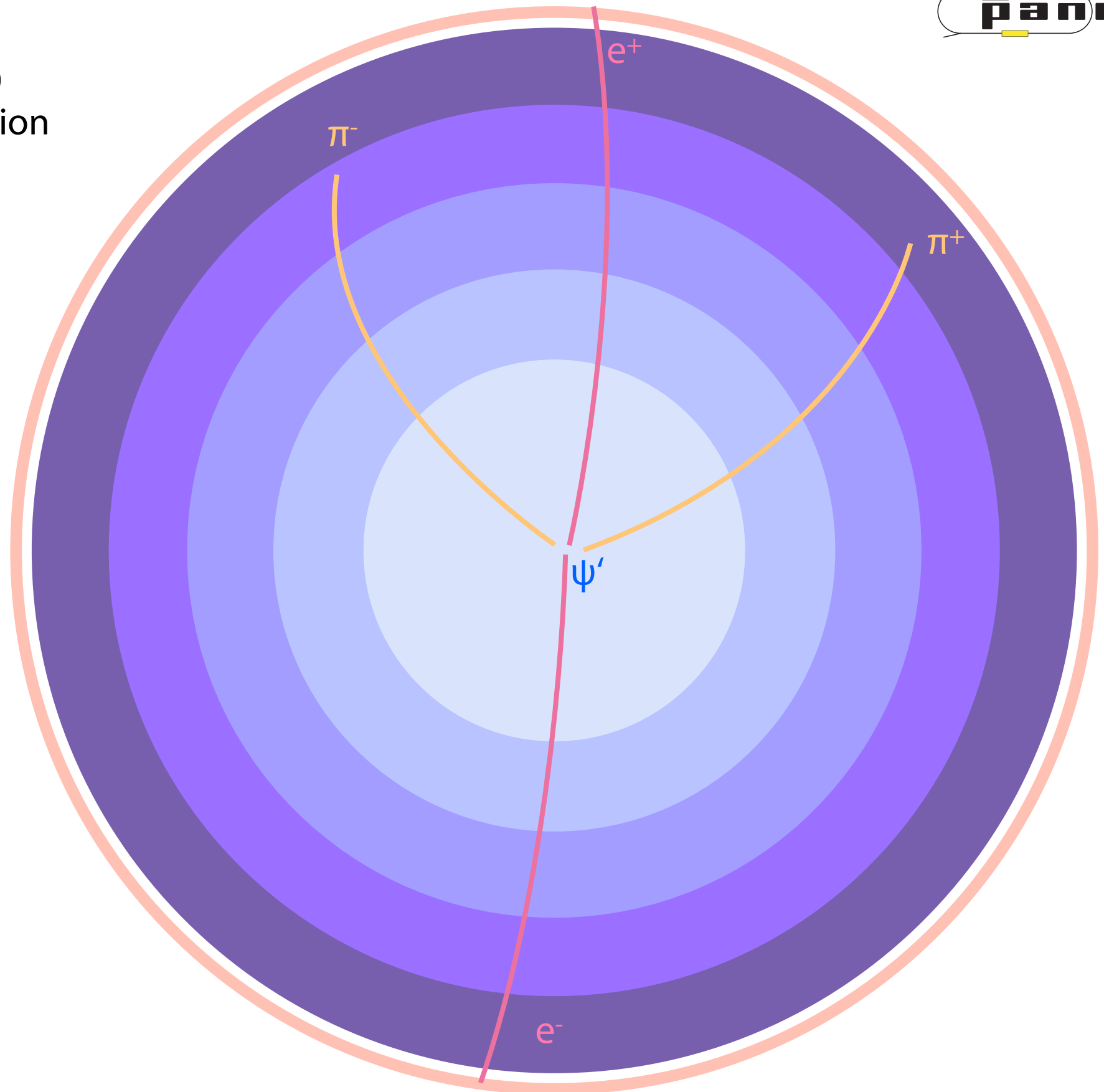
Fast detector layer(s)
Trigger data acquisition



$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

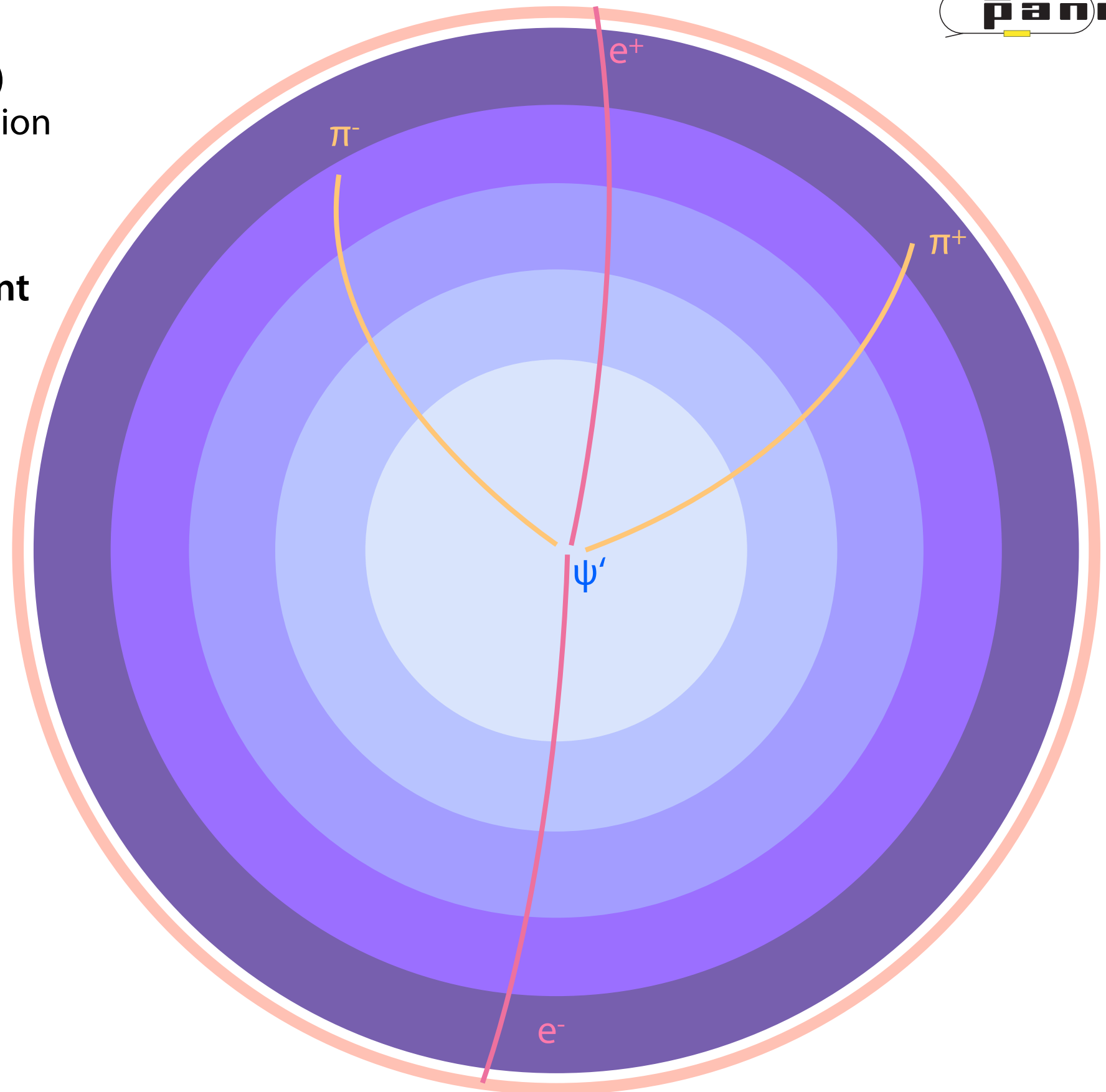


$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment

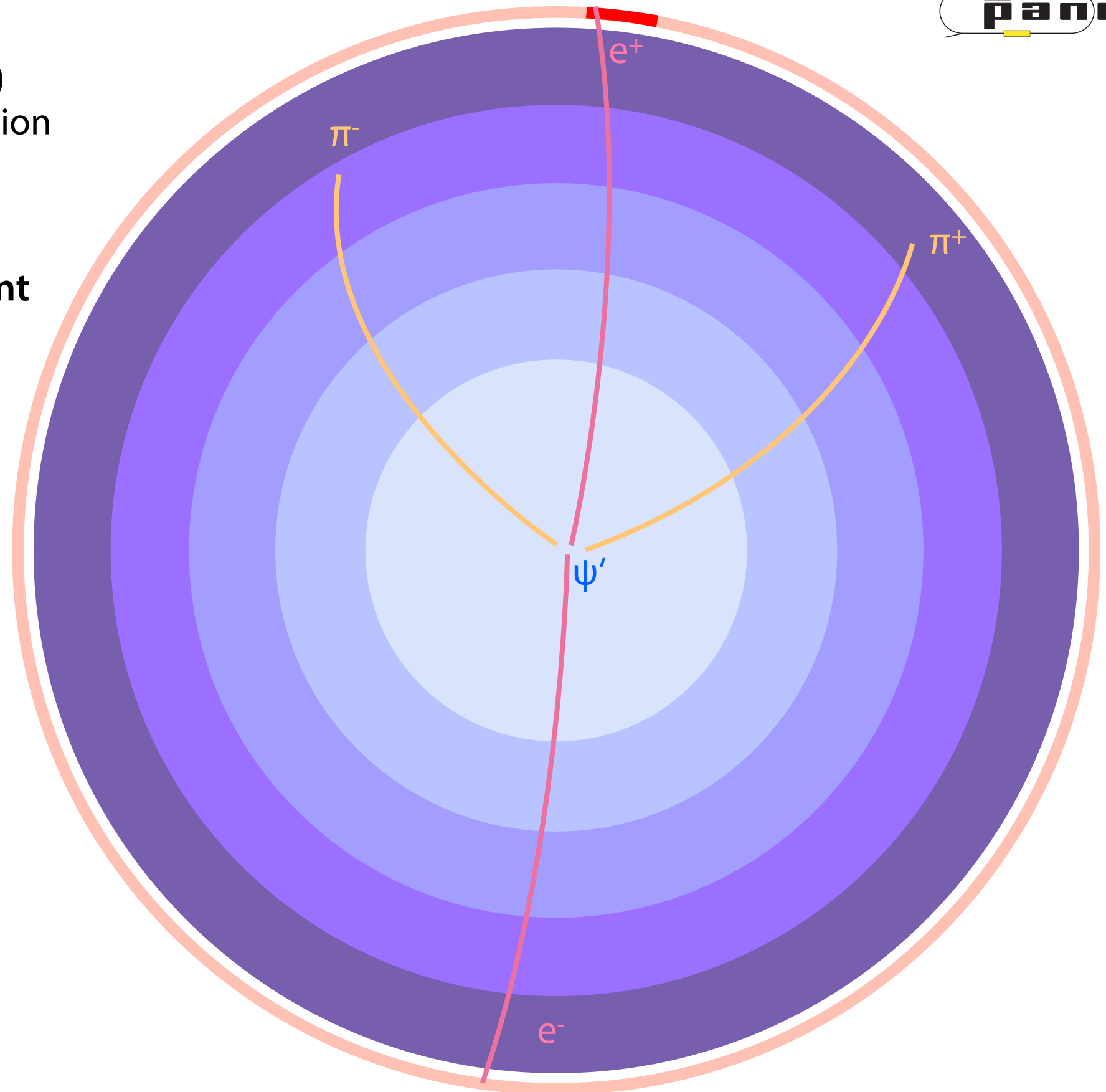


$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment

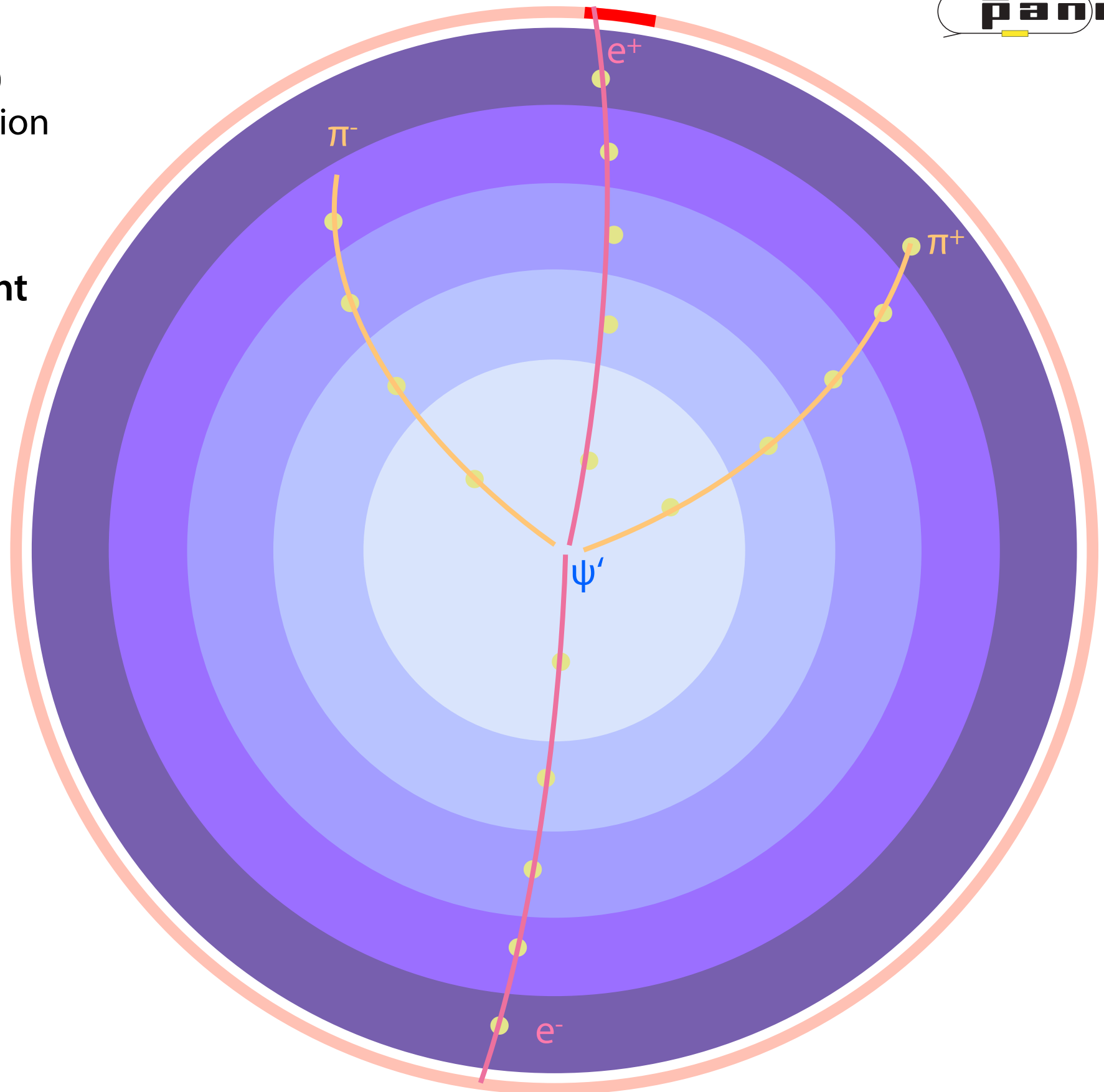


$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment

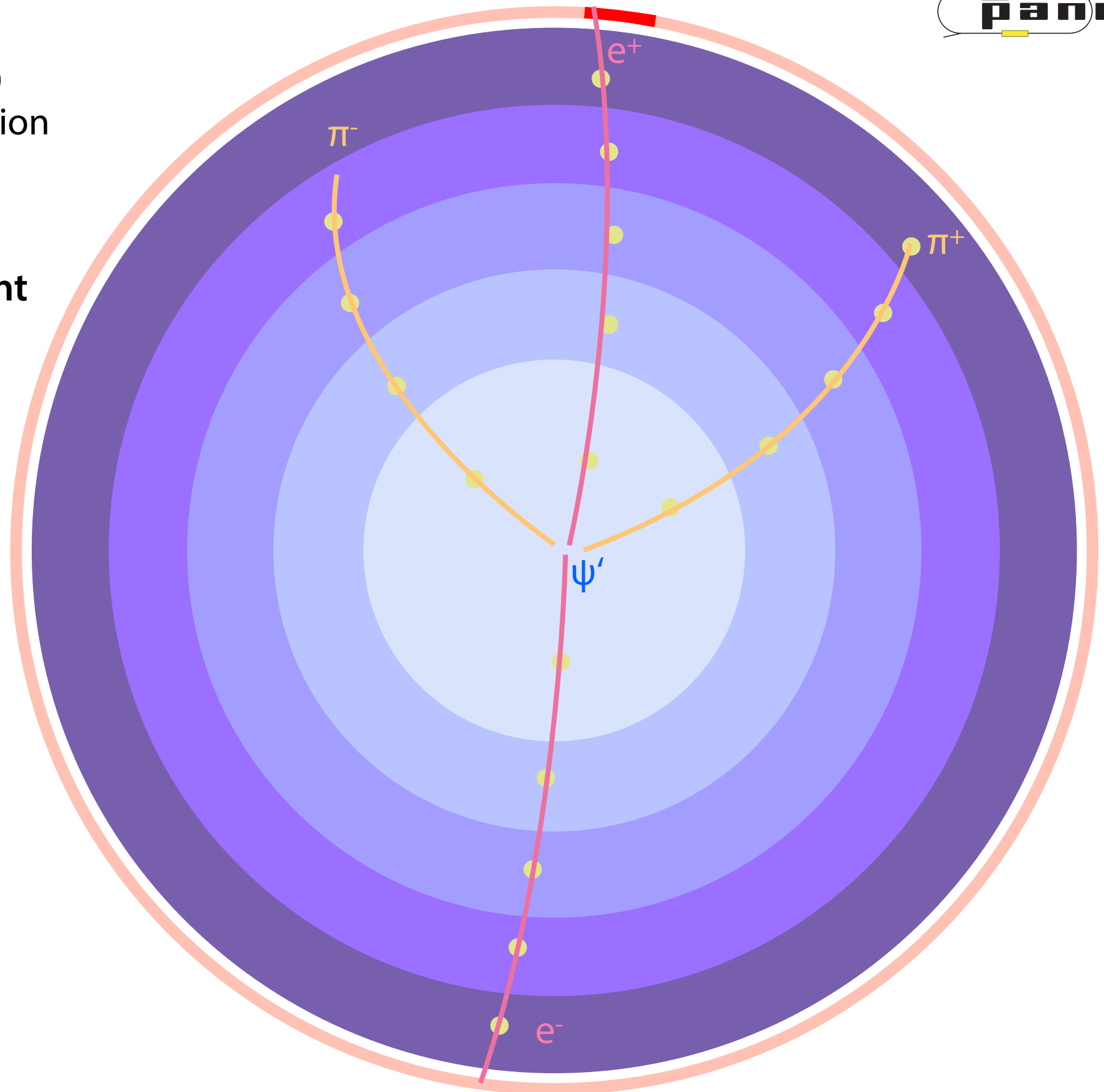


$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$ — Triggering

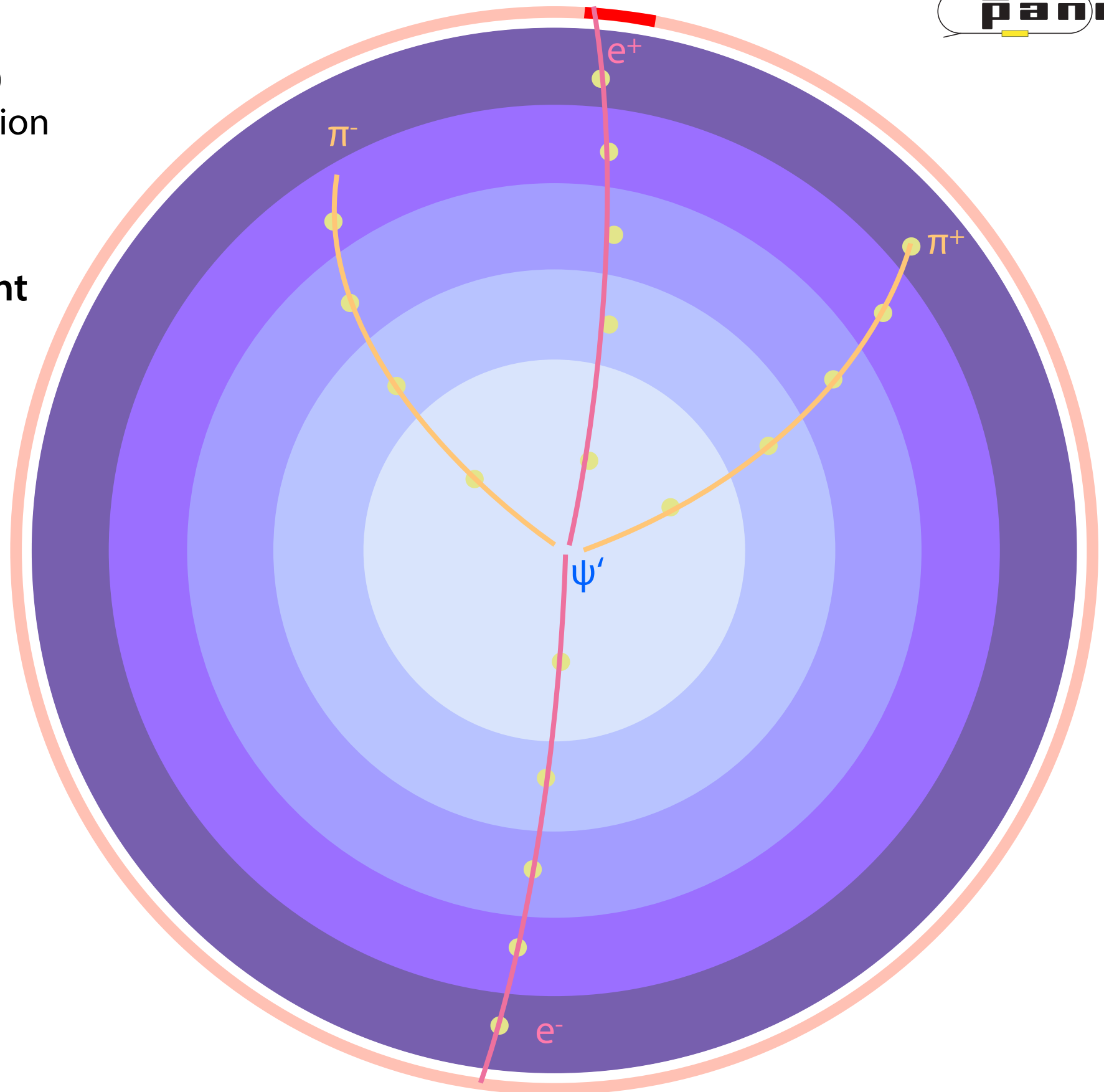
Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$



$\bar{\text{P}}\text{ANDA}$ — Triggering

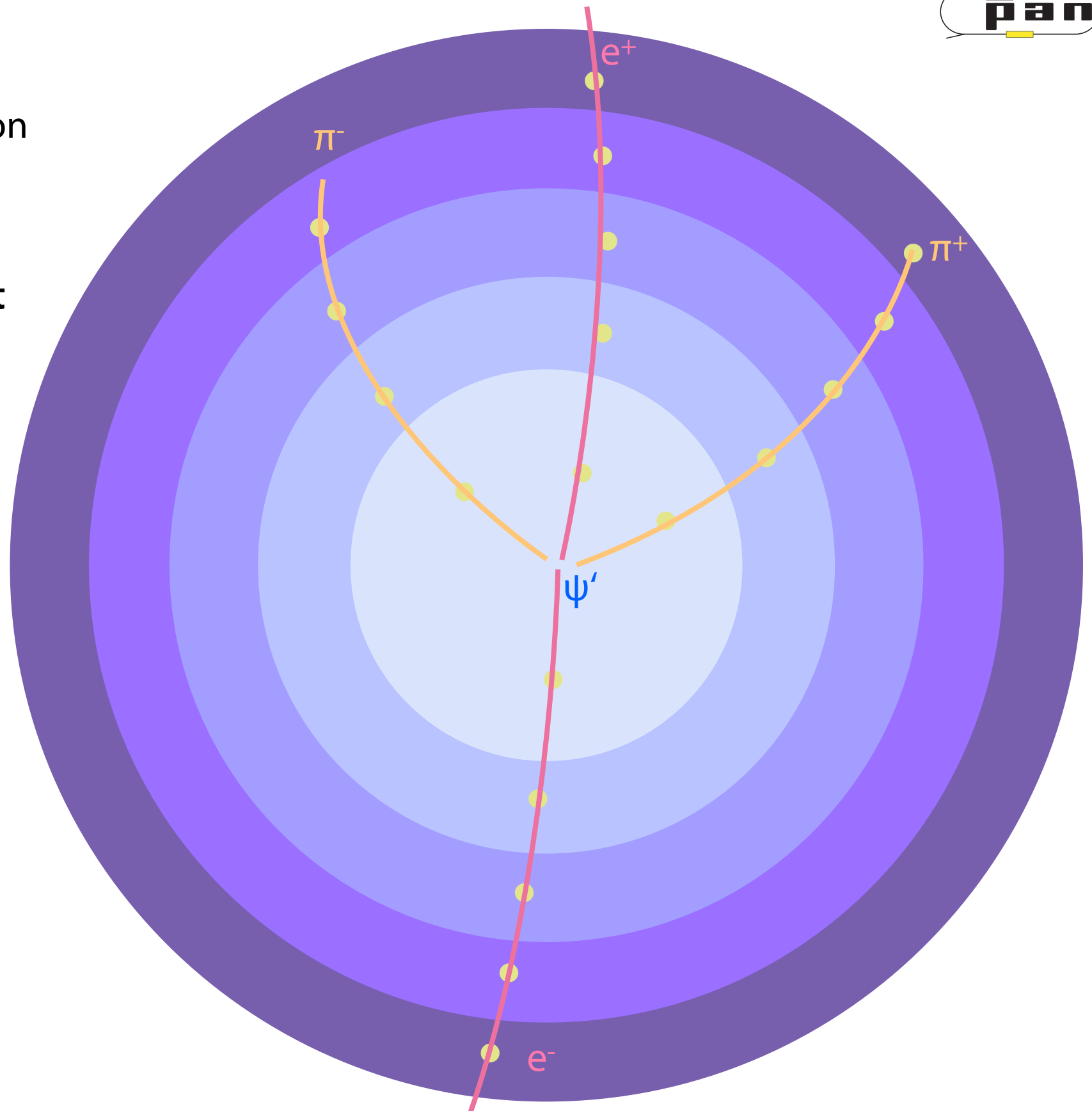
Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$



$\bar{\text{P}}\text{ANDA}$ — Triggering

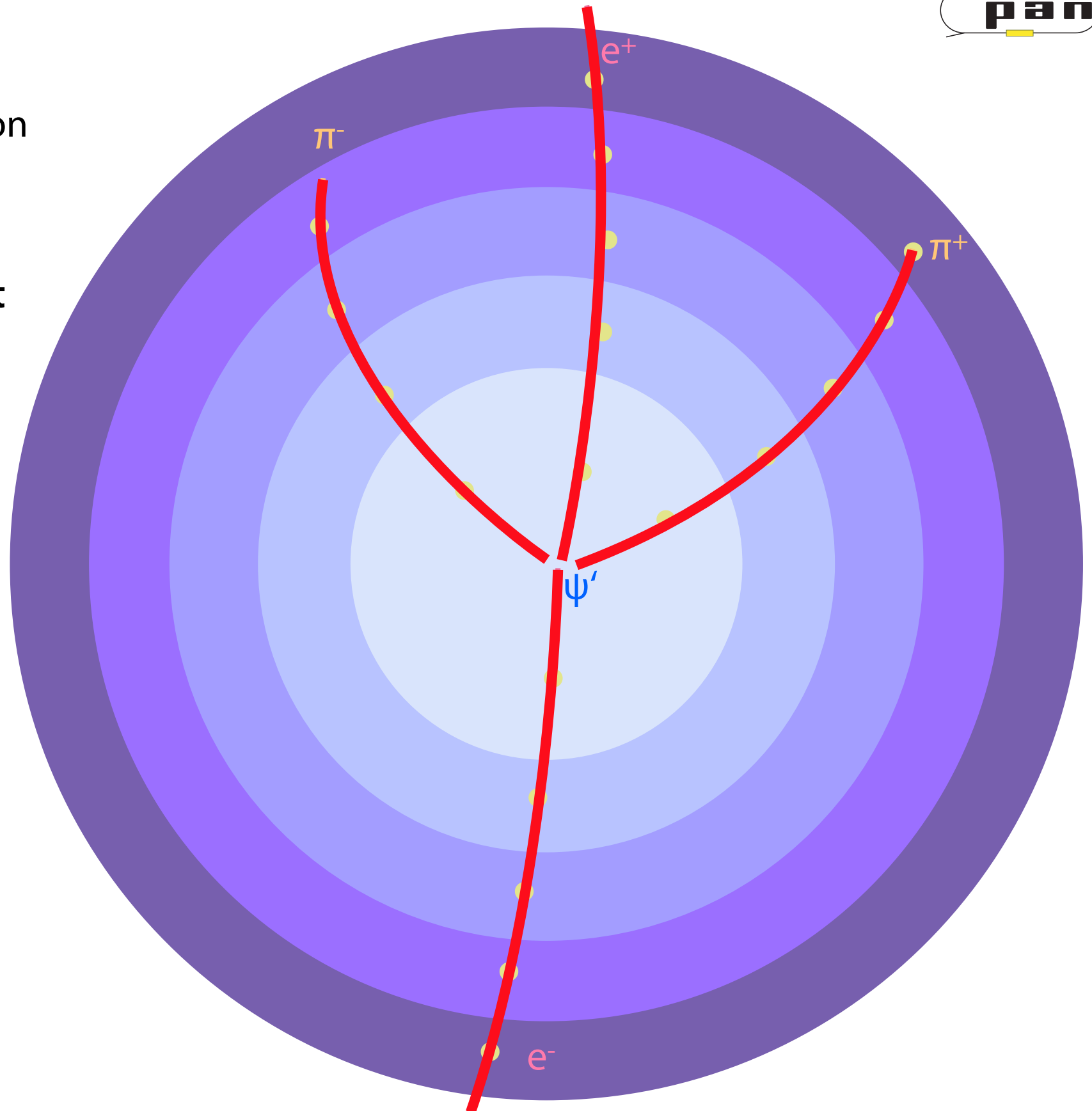
Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$

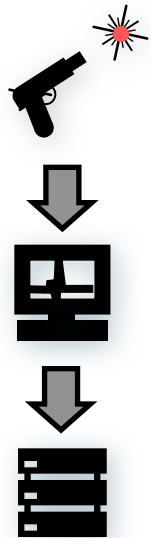


$\bar{\text{P}}\text{ANDA}$ — Triggering

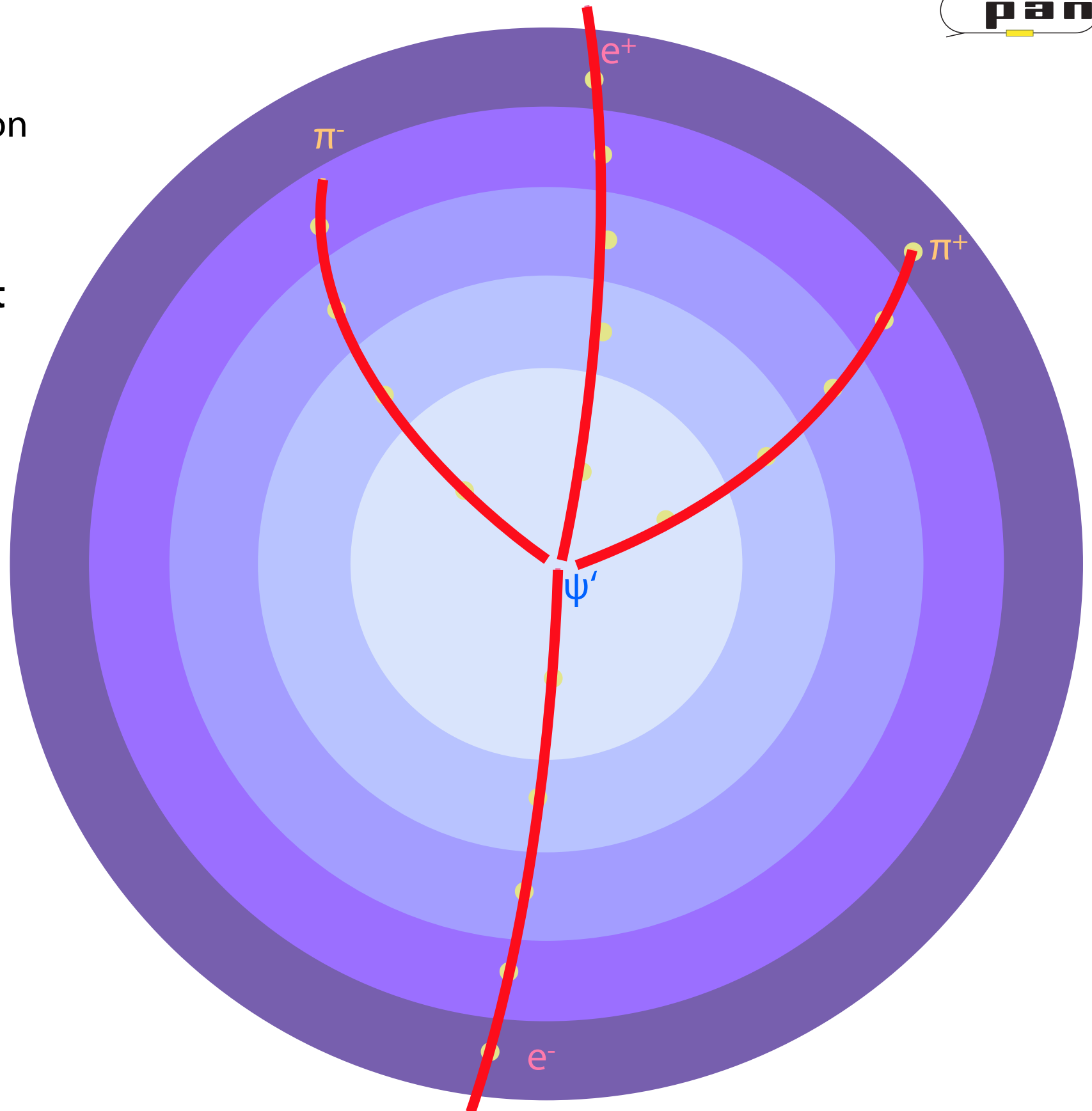
Trigger

Fast detector layer(s)
Trigger data acquisition

Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$

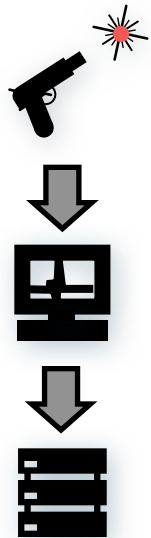


$\bar{\text{P}}\text{ANDA}$ — Triggering

Trigger

Fast detector layer(s)
Trigger data acquisition

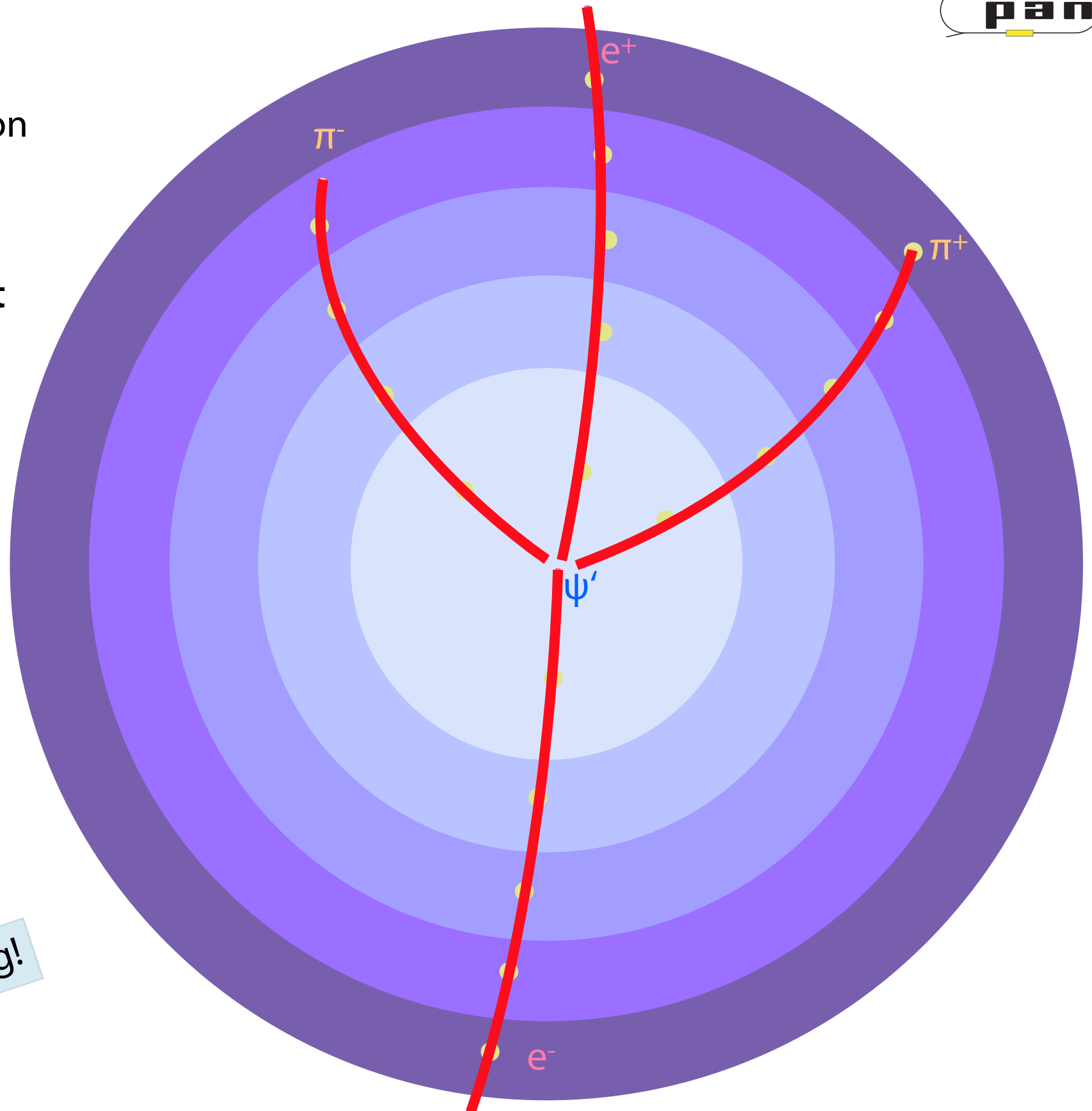
Usual HEP experiment



$\bar{\text{P}}\text{ANDA}$



Online Tracking!



ALGORITHMS

ALGORITHMS #1

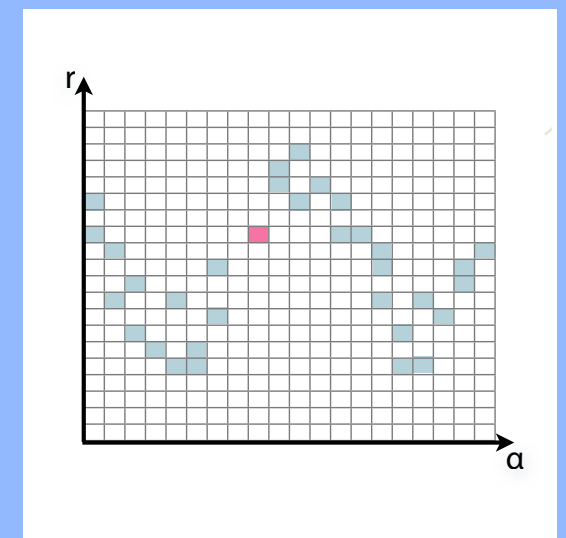
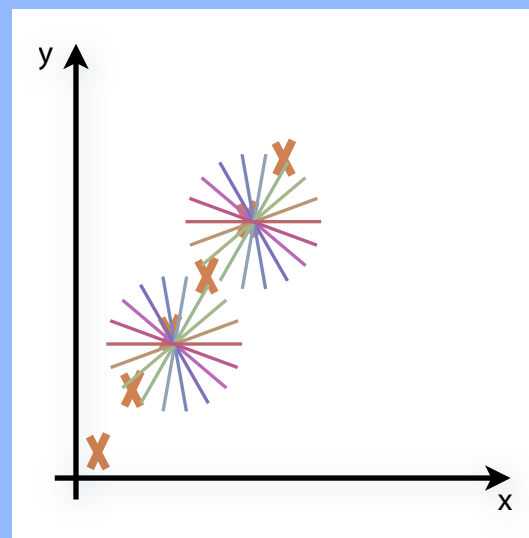
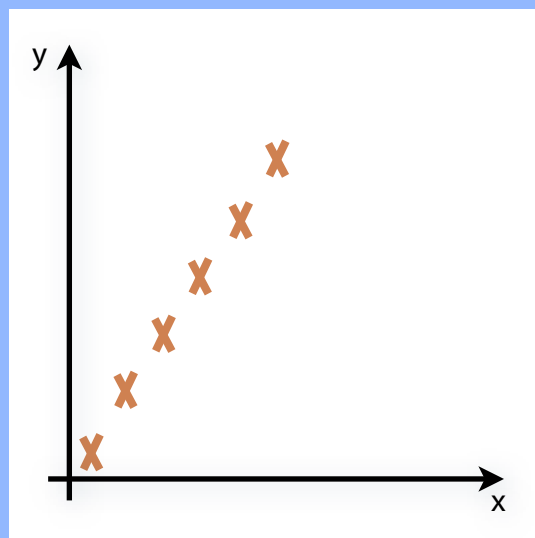
Hough Transform

Riemann Track Finder

Triplet Finder

Algorithm: Hough Transform

- **Idea:** Transform $(x,y)_i \rightarrow (\alpha,r)_{ij}$, find lines via (α,r) space
- Solve r_{ij} line equation for
 - Lots of hits $(x,y,\rho)_i$ and
 - Many $\alpha_j \in [0^\circ, 360^\circ)$ each
- Fill histogram
- Extract track parameters



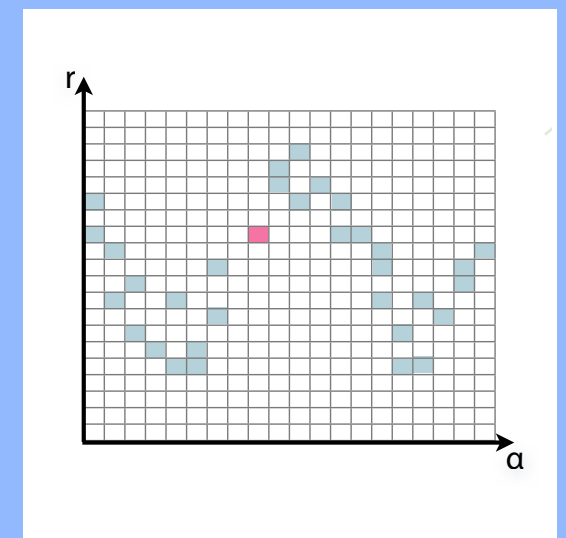
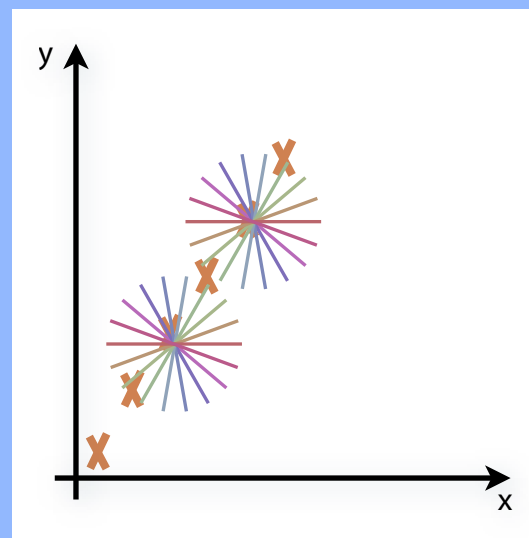
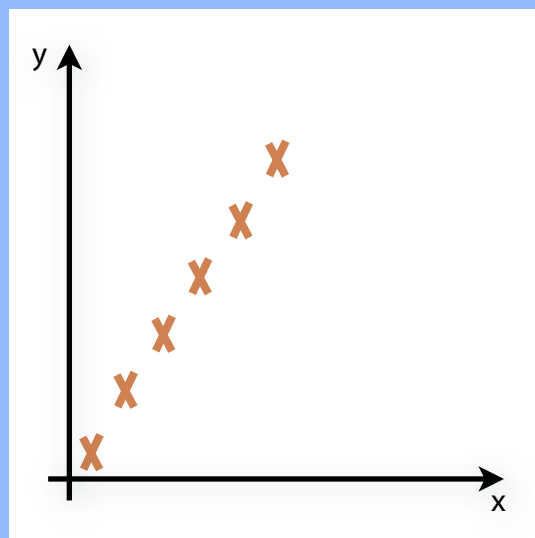
Algorithm: Hough Transform

- **Idea:** Transform $(x,y)_i \rightarrow (\alpha,r)_{ij}$, find lines via (α,r) space
- Solve r_{ij} line equation for
 - Lots of hits $(x,y,\rho)_i$ and
 - Many $\alpha_j \in [0^\circ, 360^\circ)$ each
- Fill histogram
- Extract track parameters

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$

i: ~100 hits/event (STT)
j: every 0.2°

r_{ij} : 180 000



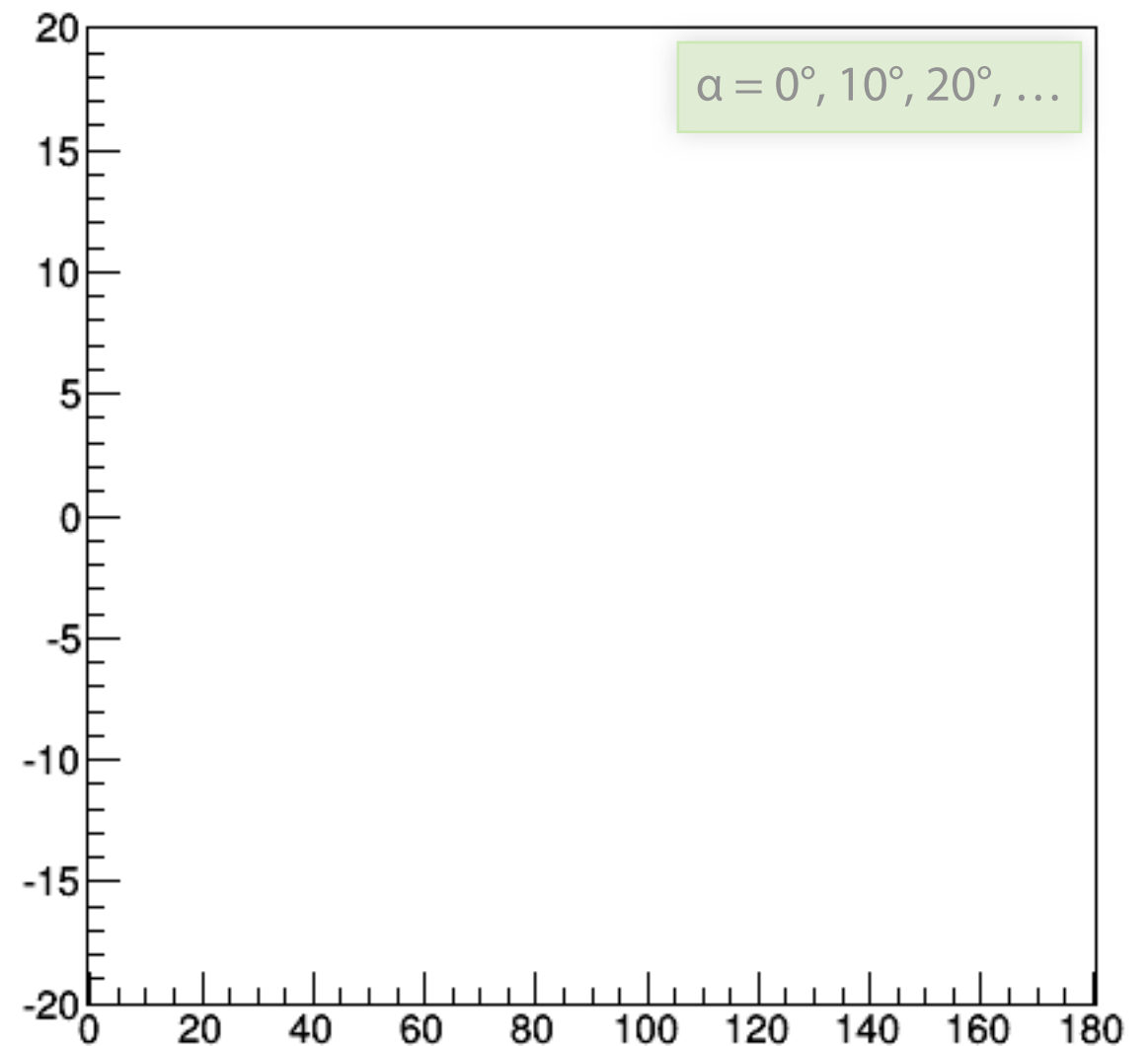
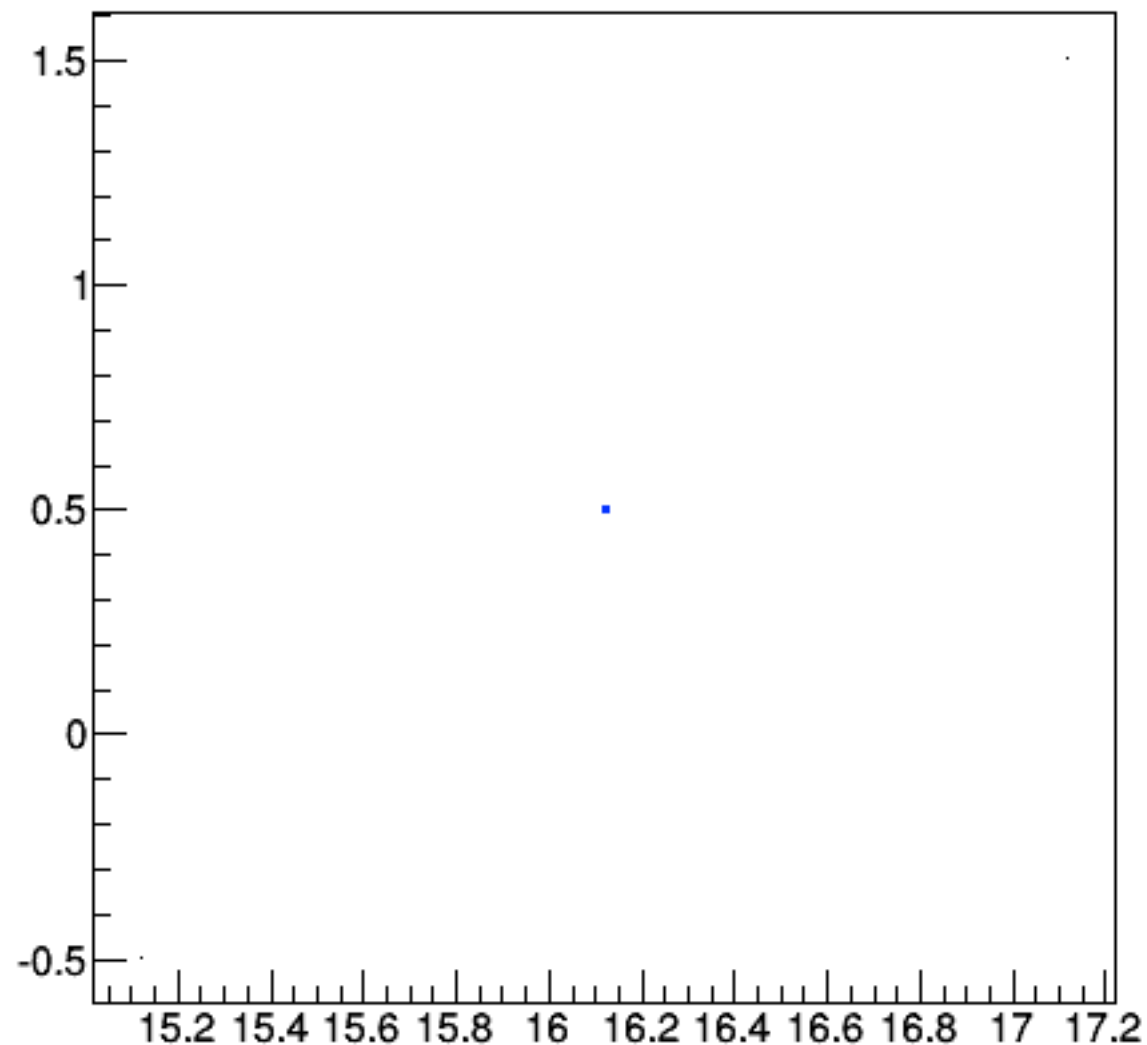
Hough Transform — Visualization Lines

- Create lines going through hit point $(x,y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ)$

$\alpha = 0^\circ, 10^\circ, 20^\circ, \dots$

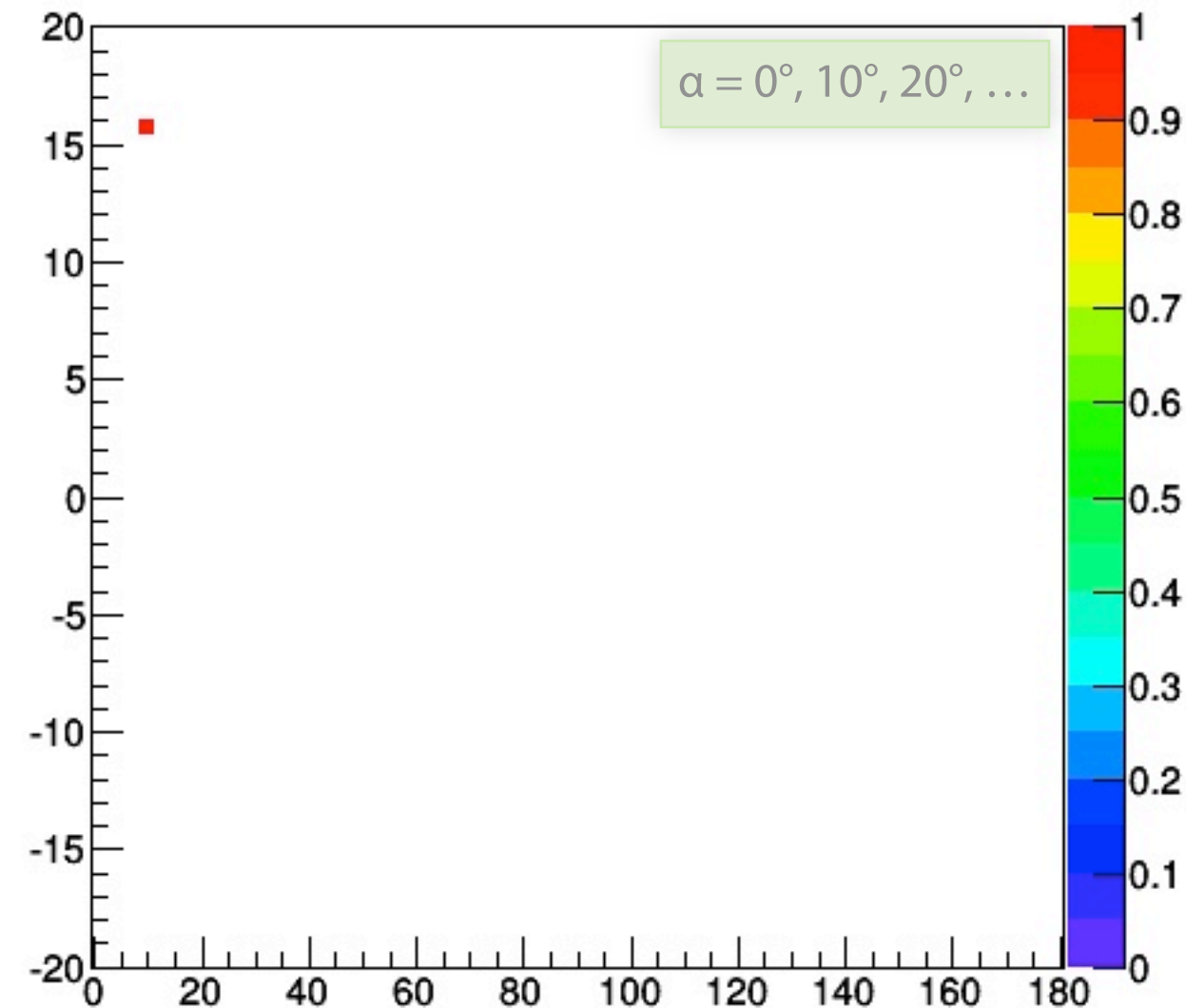
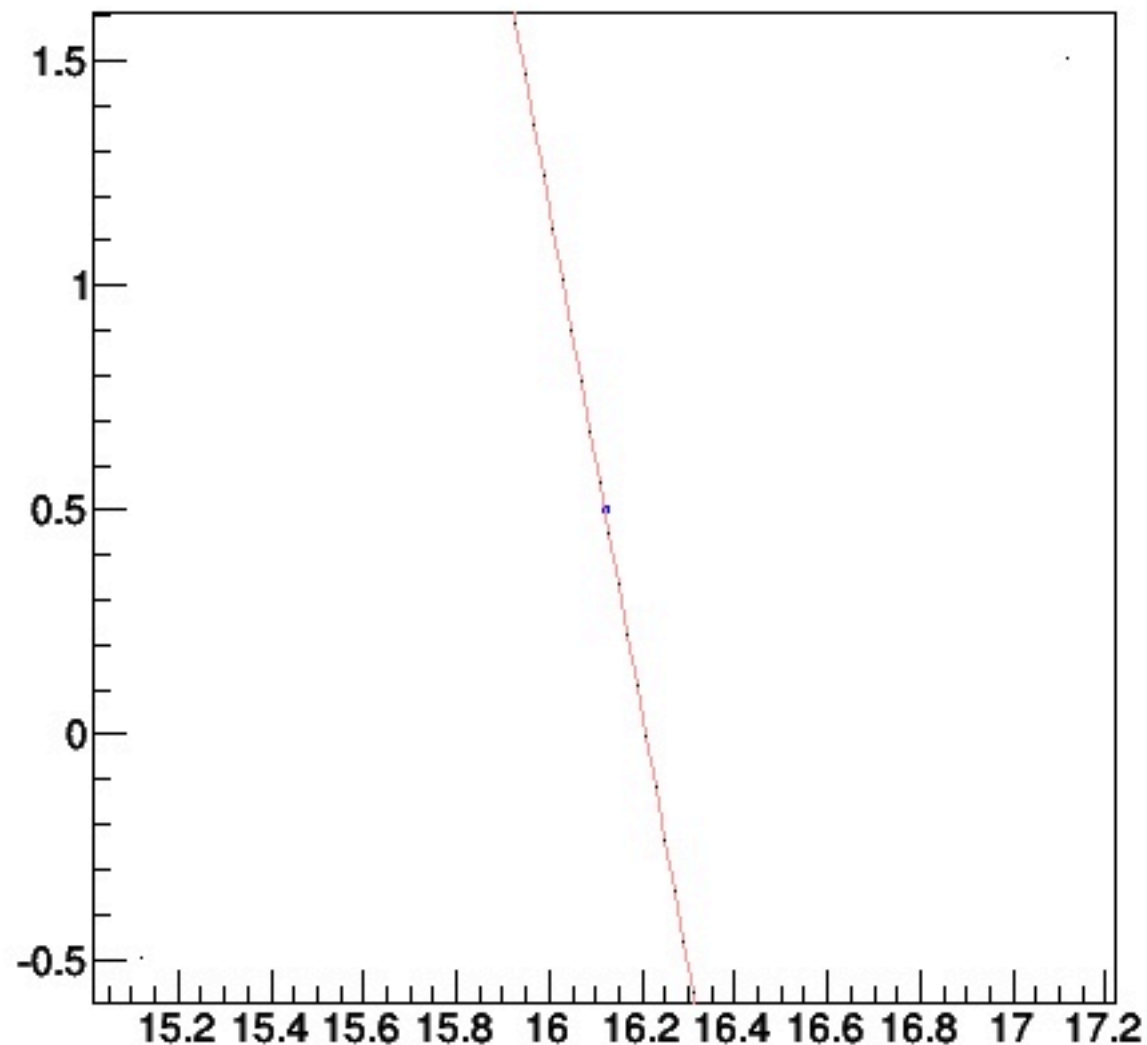
Hough Transform — Visualization Lines

- Create lines going through hit point $(x, y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ)$



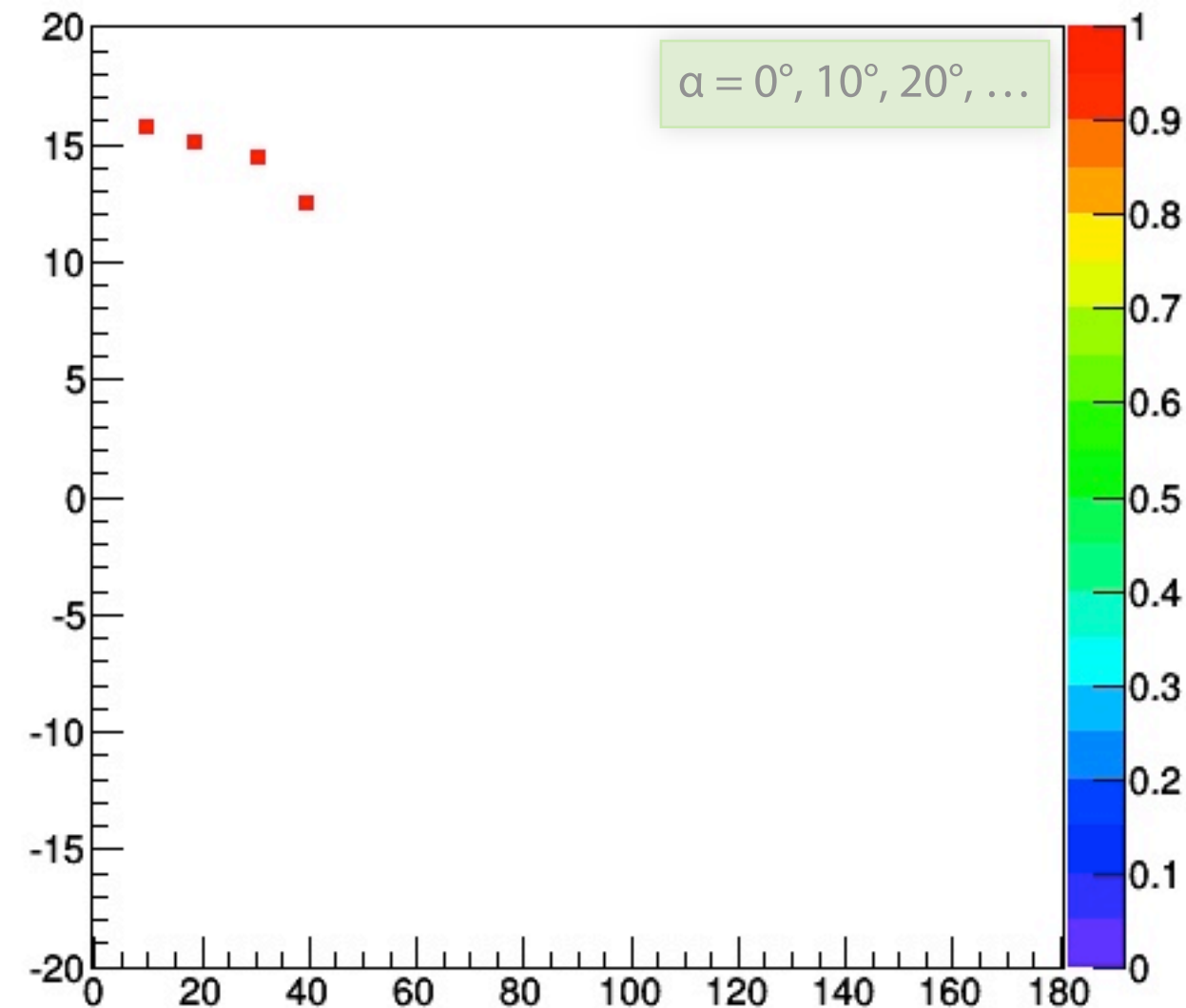
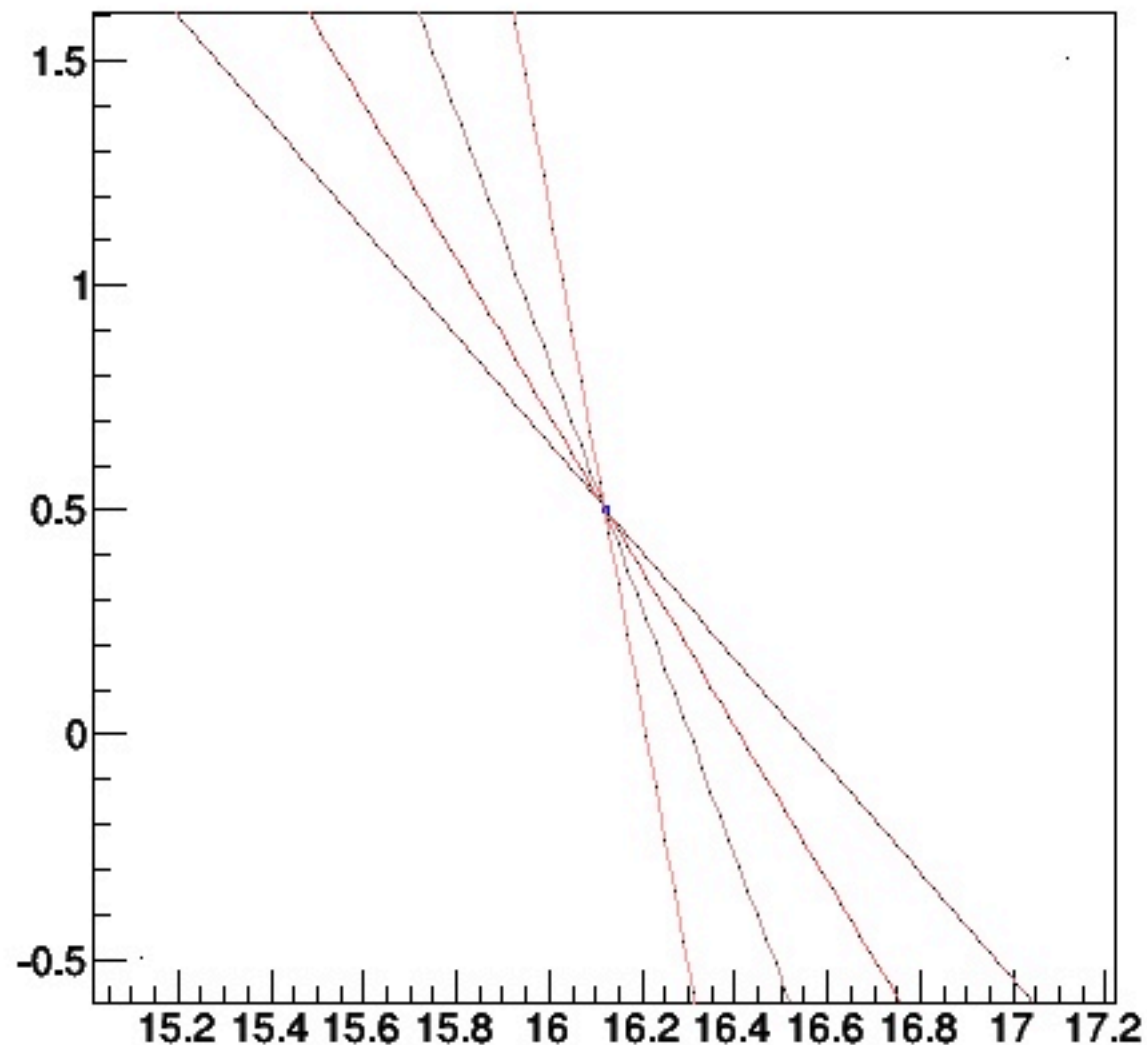
Hough Transform — Visualization Lines

- Create lines going through hit point $(x, y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ)$



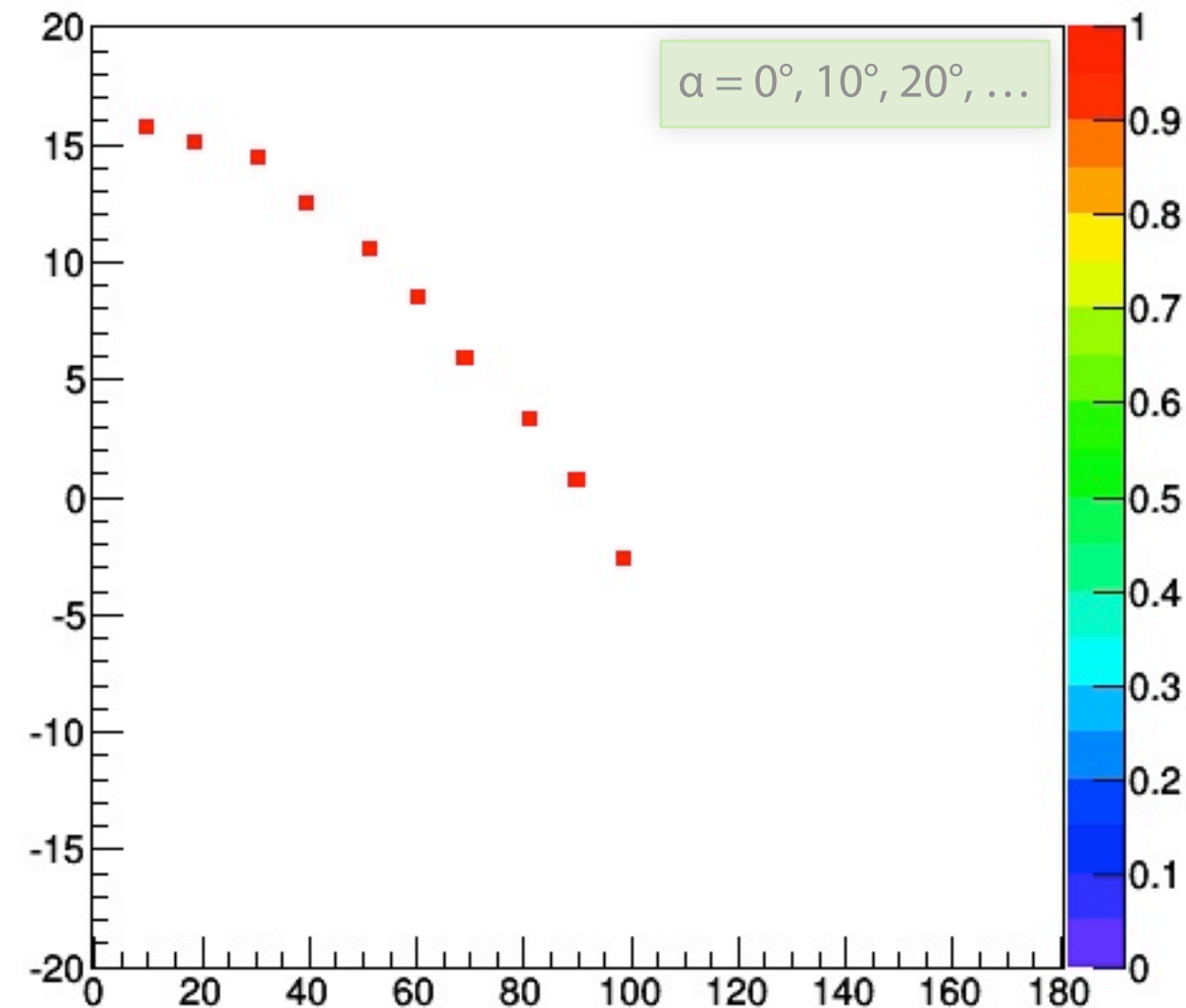
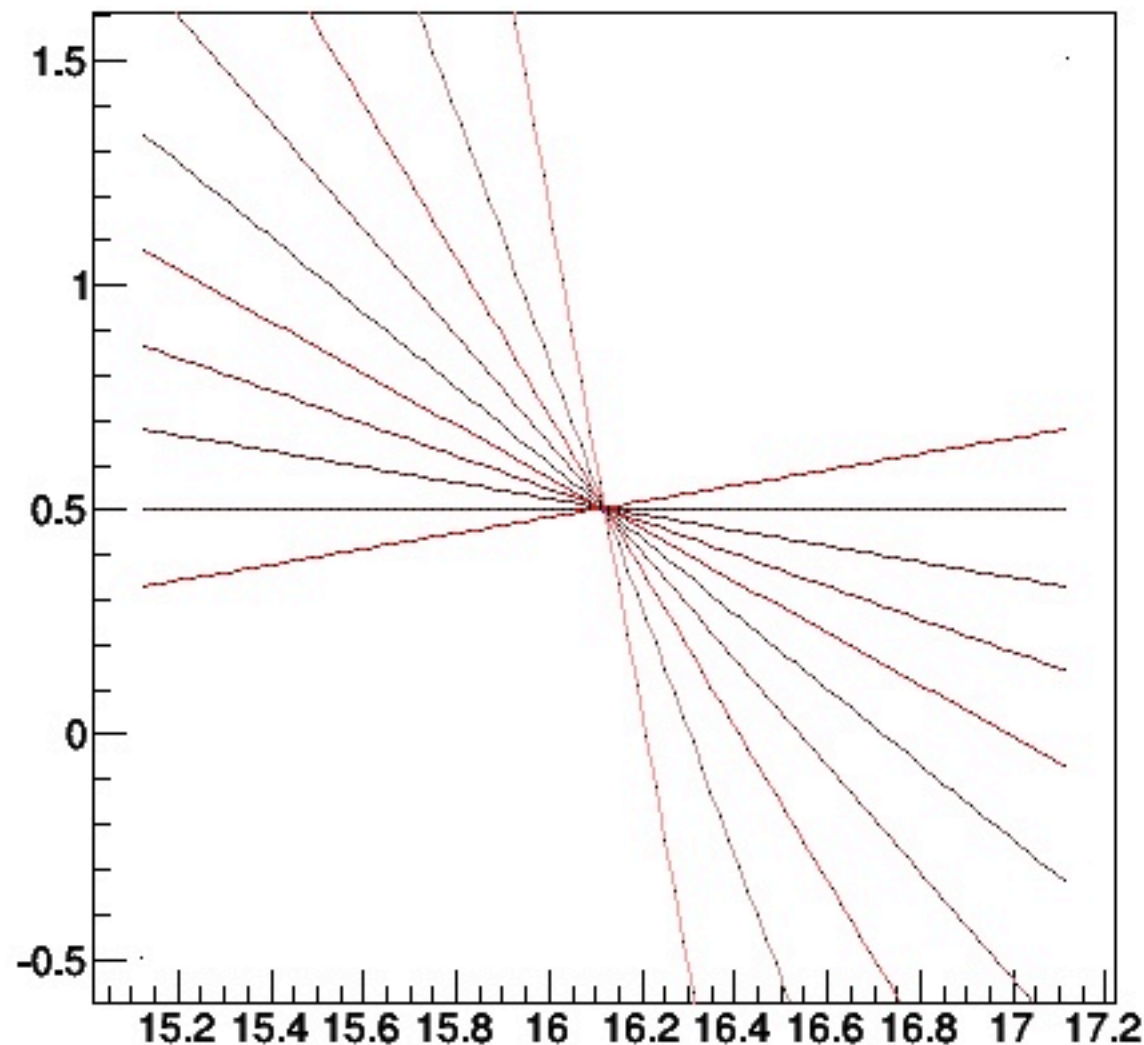
Hough Transform — Visualization Lines

- Create lines going through hit point $(x, y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ)$



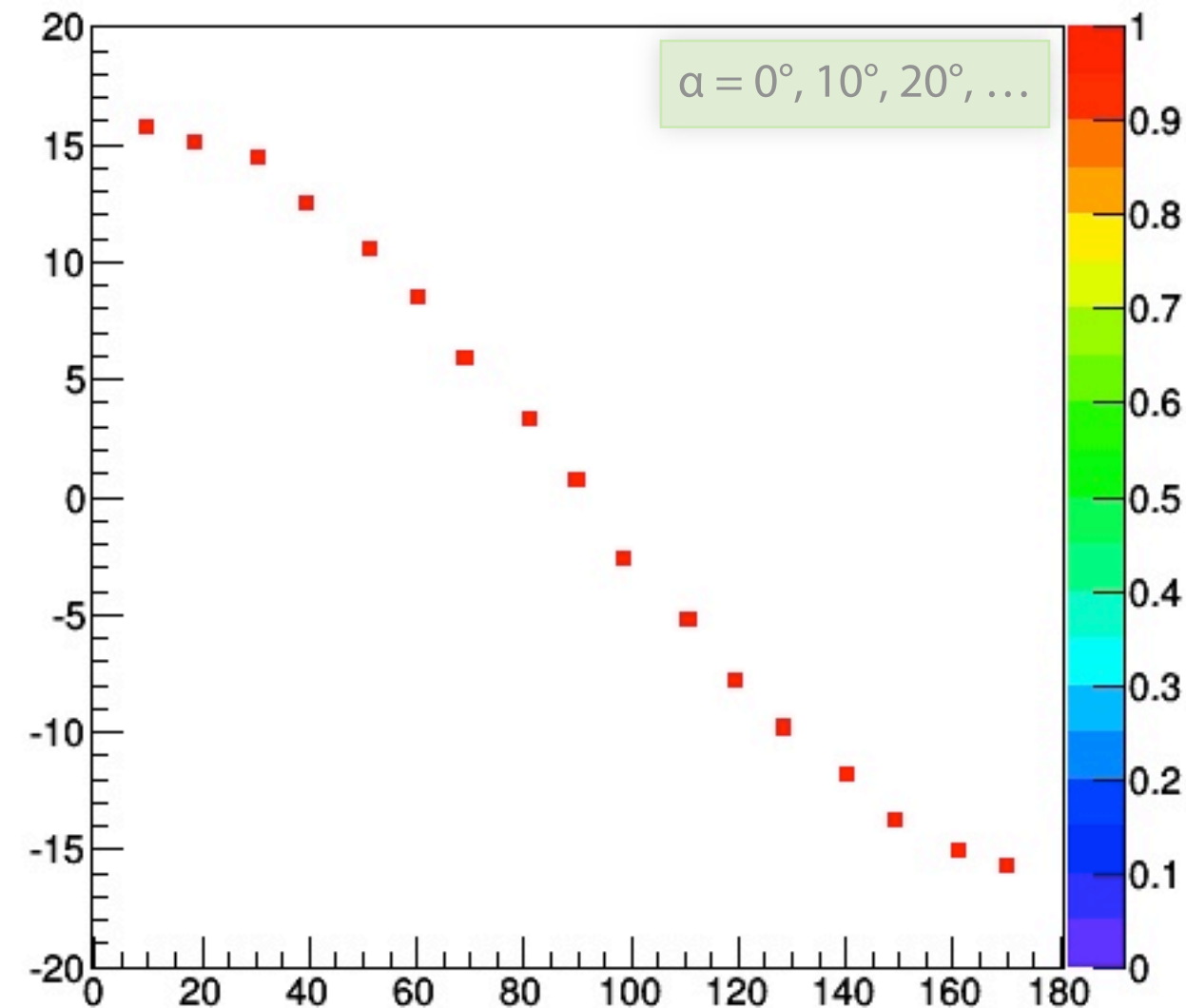
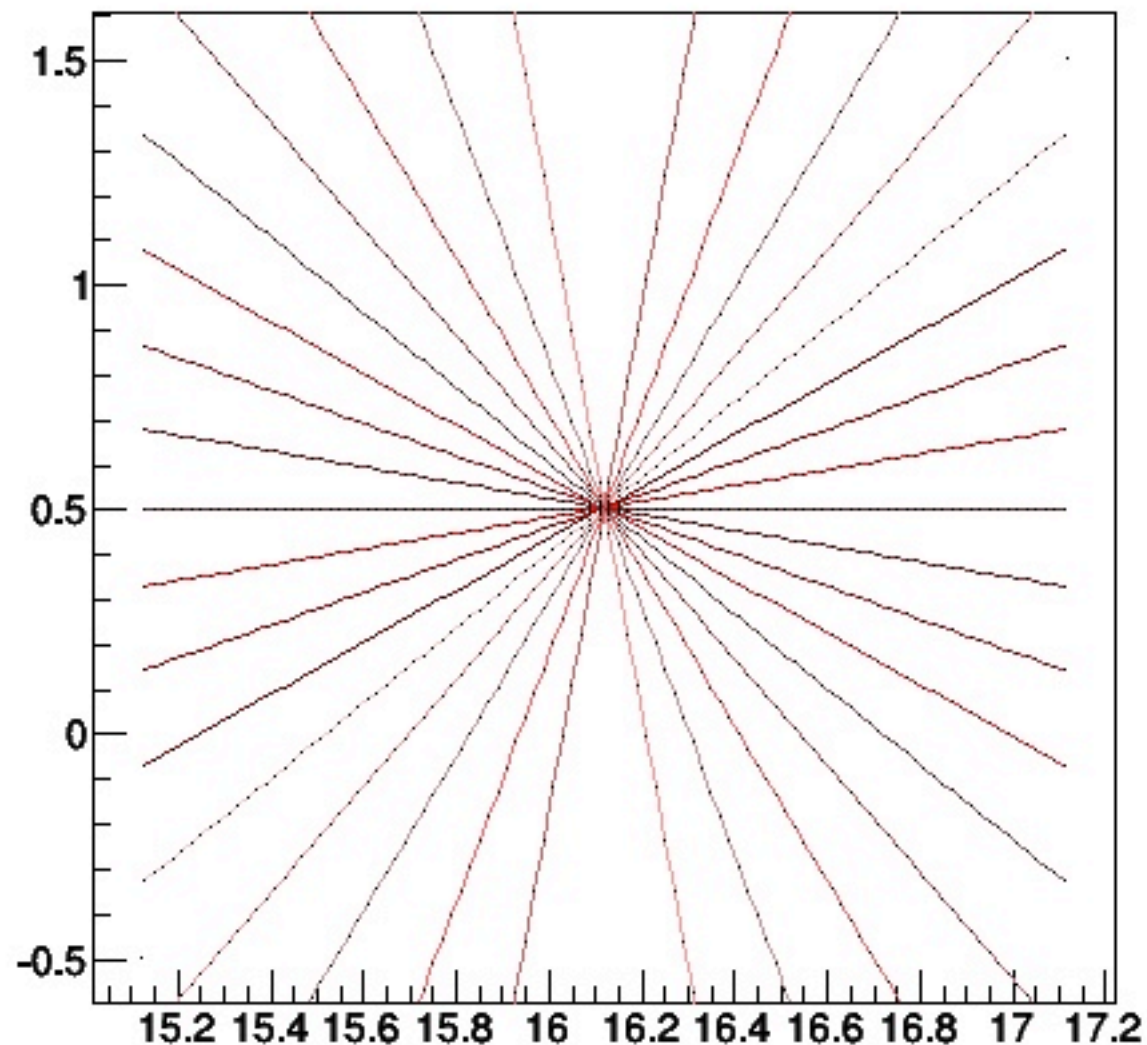
Hough Transform — Visualization Lines

- Create lines going through hit point $(x, y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ)$



Hough Transform — Visualization Lines

- Create lines going through hit point $(x, y)_i$
 - Line parameterized by $r_{ij} = \cos(\alpha_j) \cdot x_i + \sin(\alpha_j) \cdot y_i$
- Fill line parameters $(\alpha, r)_{ij}$ into histogram
 - Rasterize for many $\alpha_j \in [0^\circ, 180^\circ]$



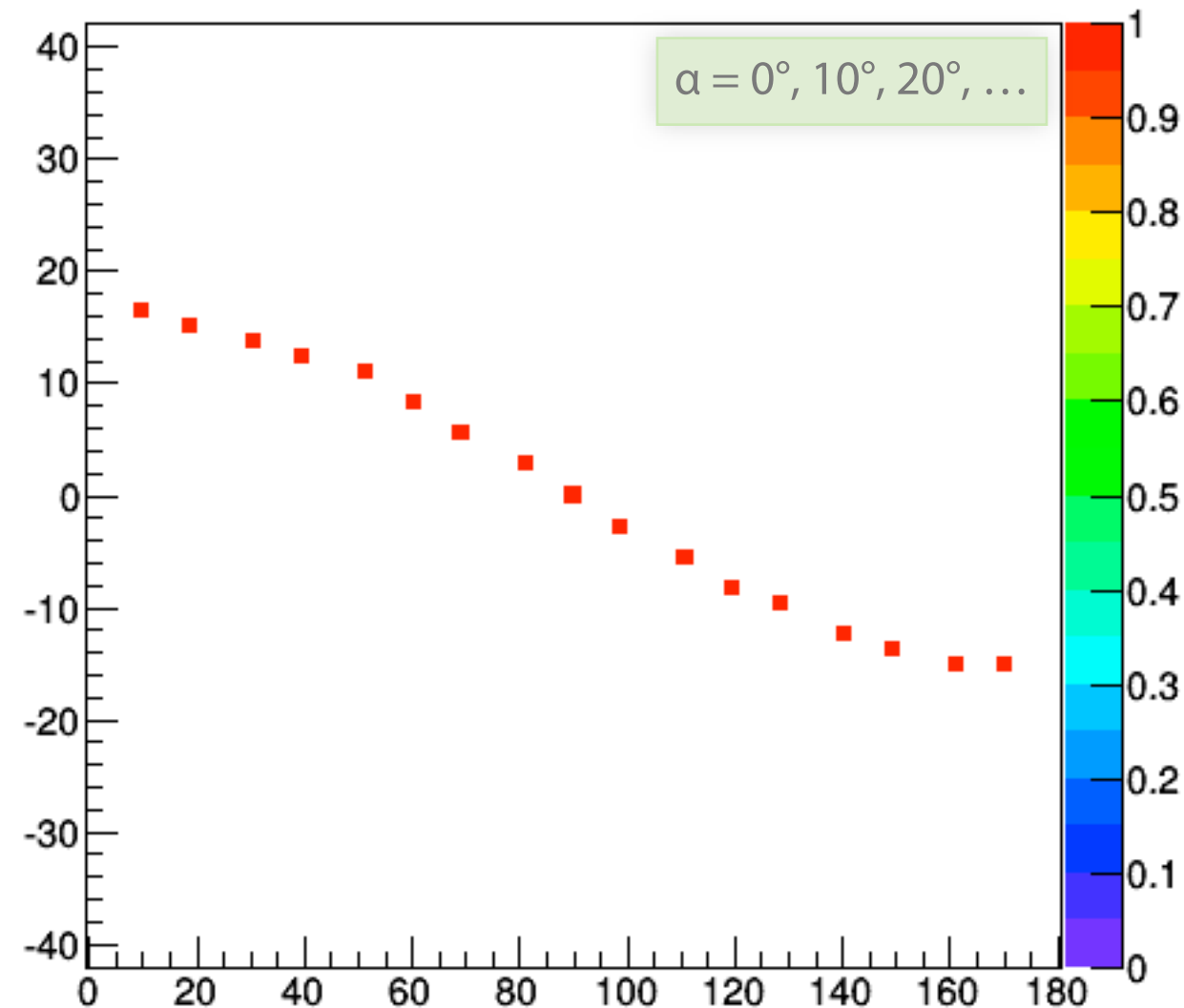
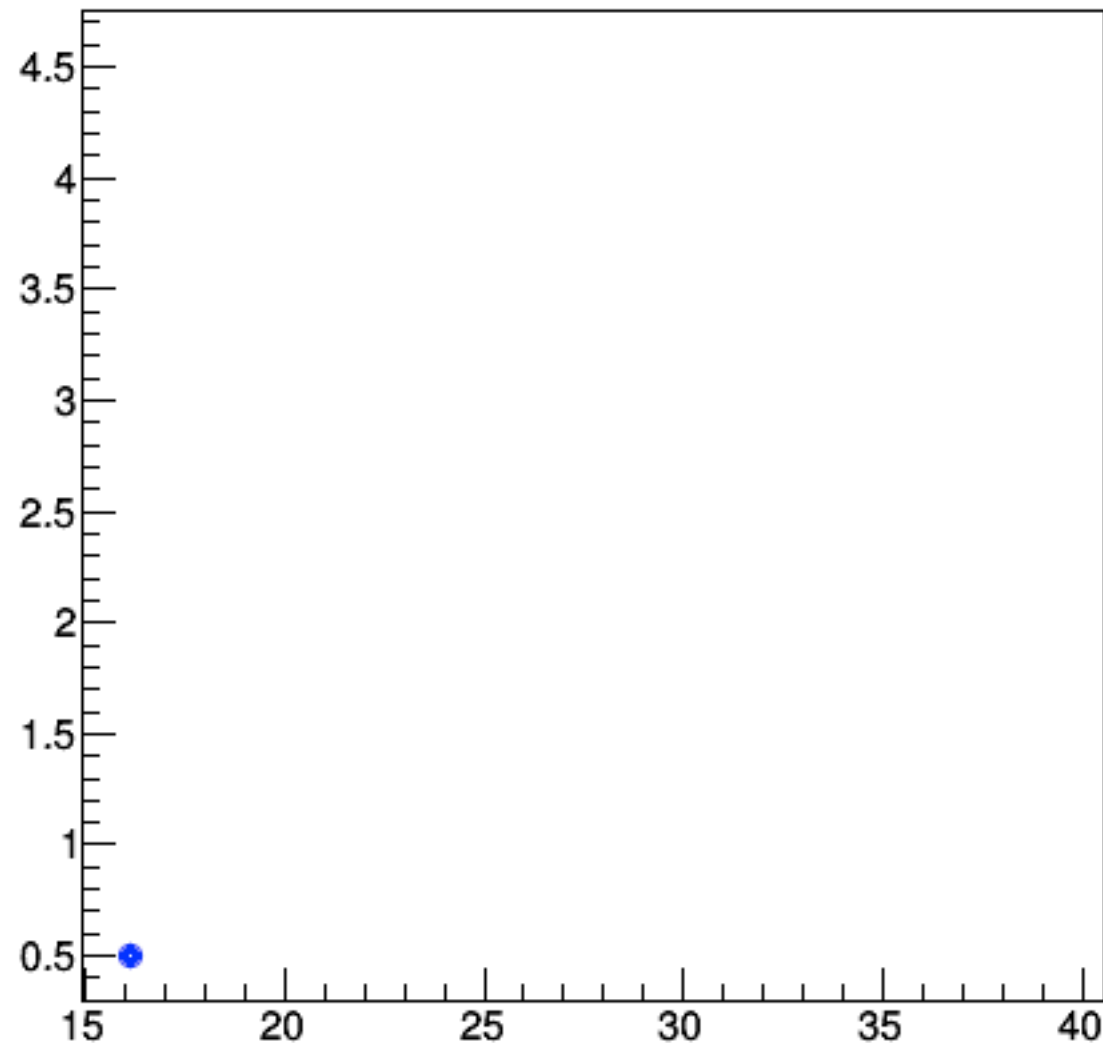
Hough Transform — Visualization Points

- Create lines going through hit point $(x,y)_i$
- **Repeat** for every hit point i

$$\alpha = 0^\circ, 10^\circ, 20^\circ, \dots$$

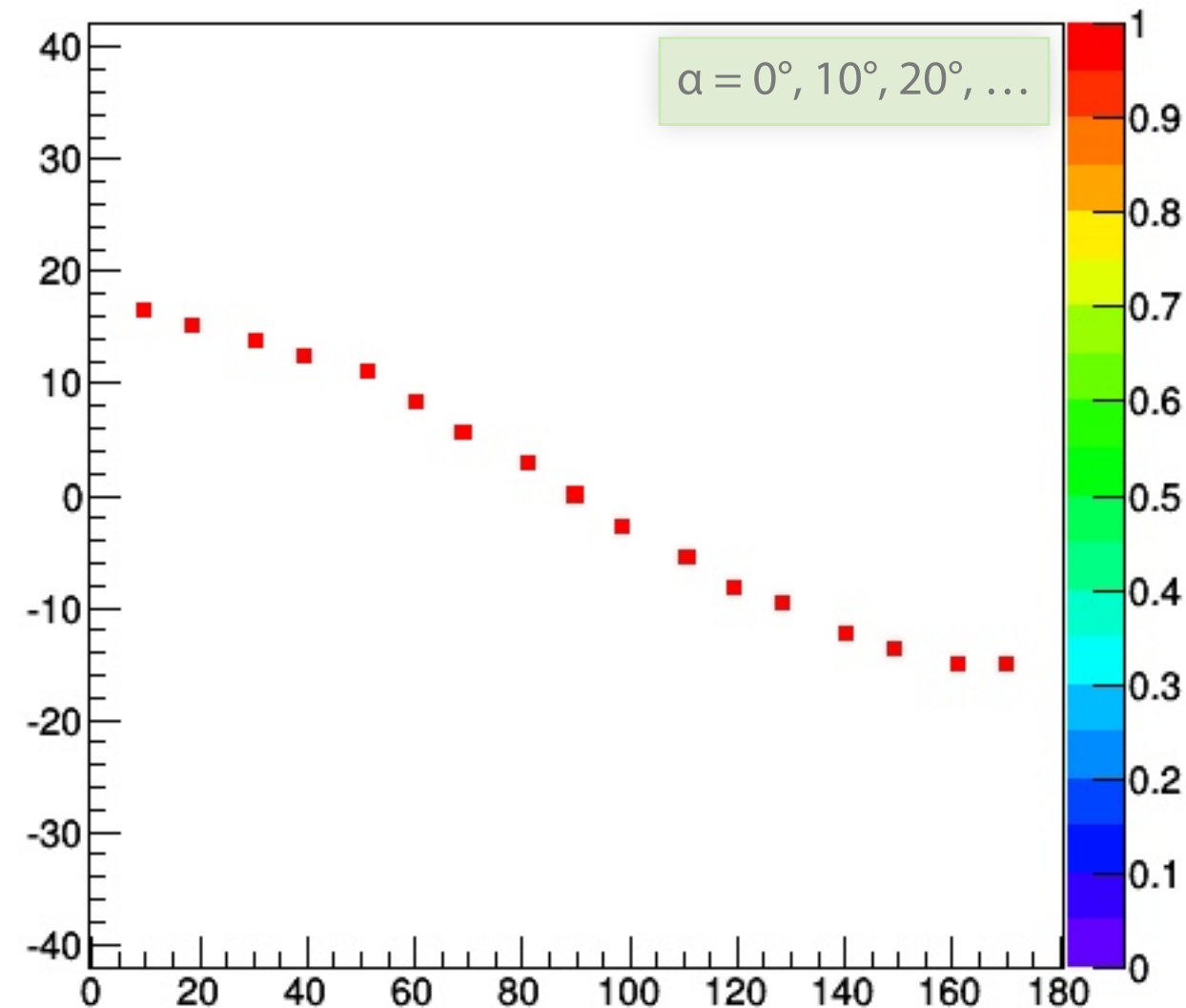
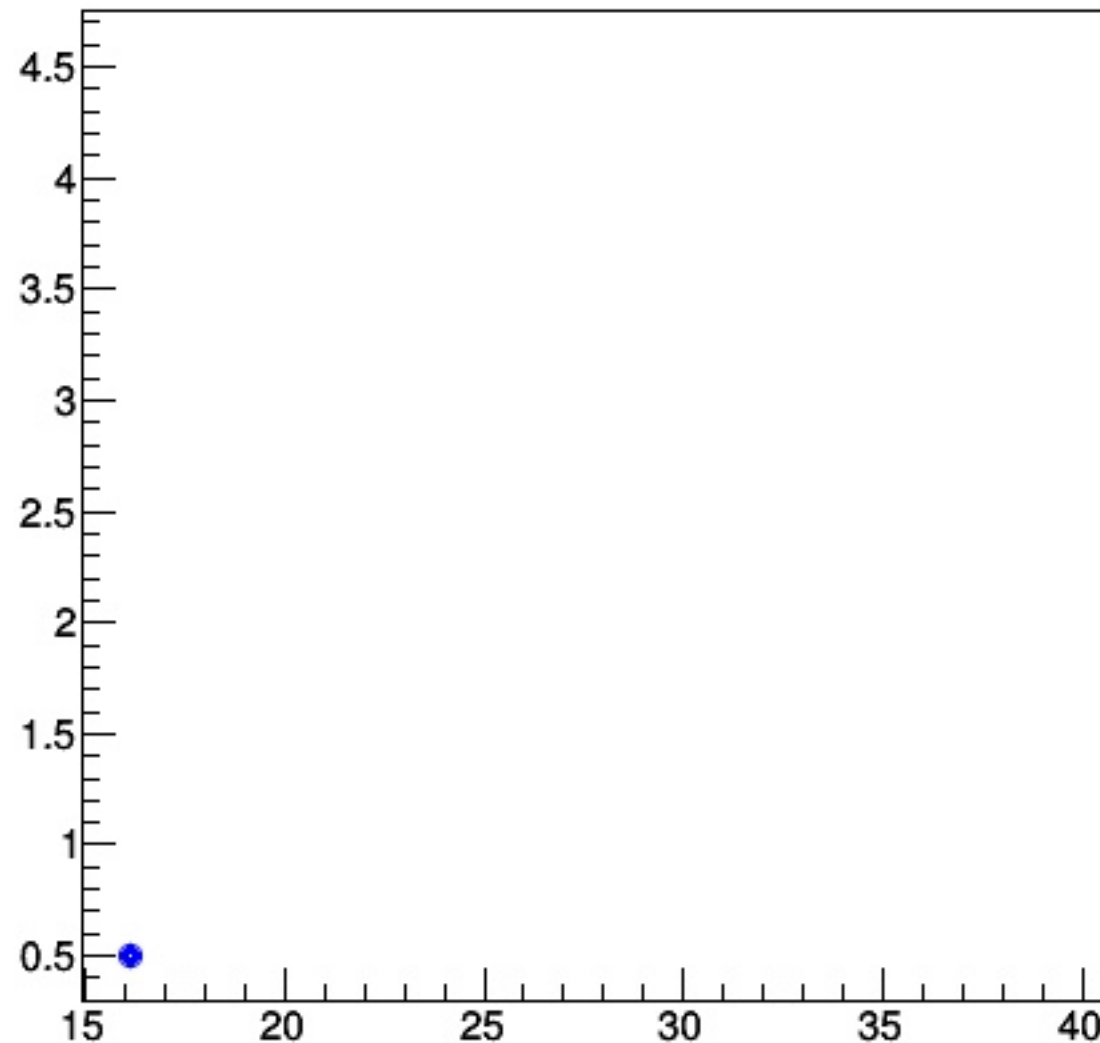
Hough Transform — Visualization Points

- Create lines going through hit point $(x, y)_i$
- **Repeat** for every hit point i



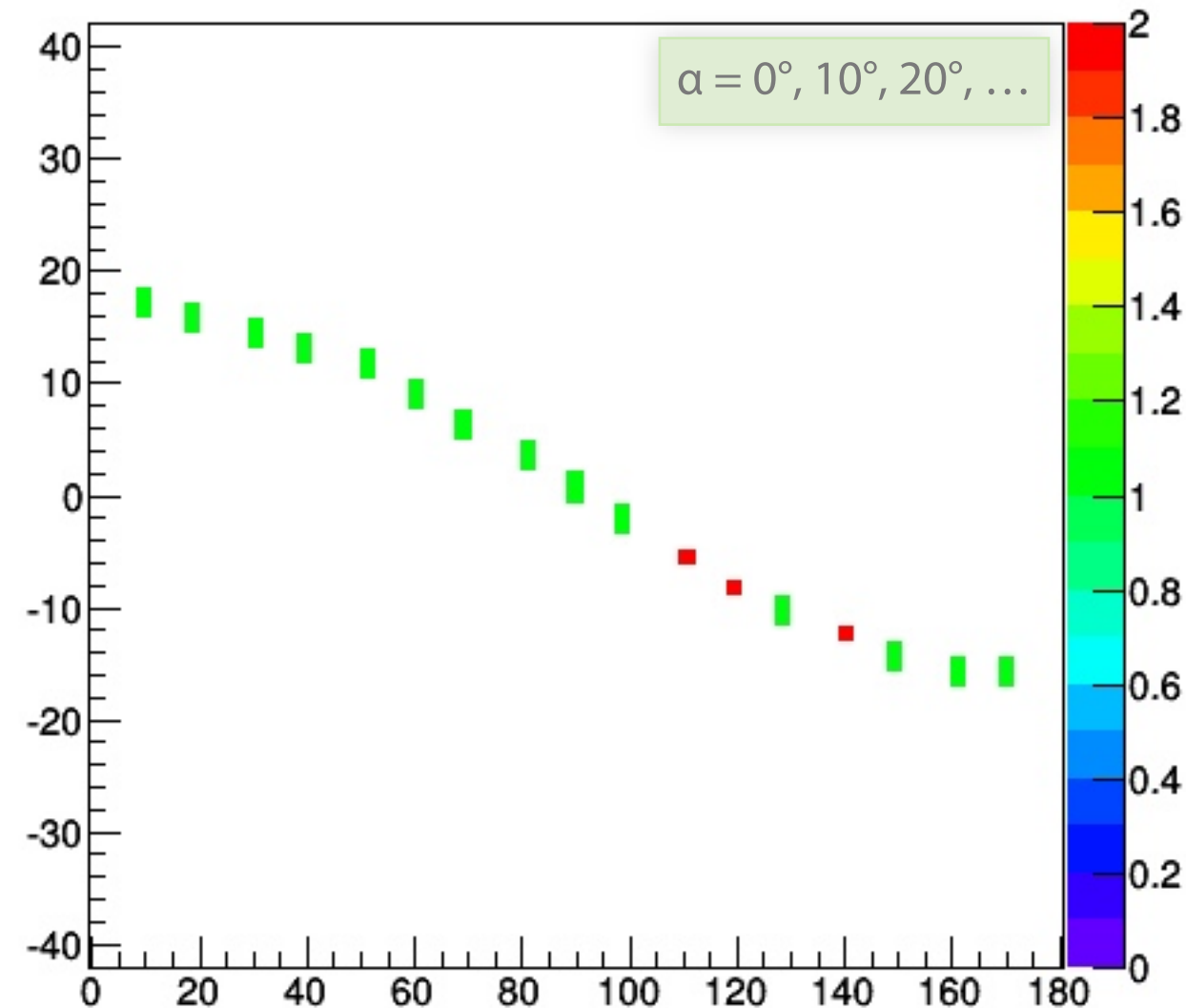
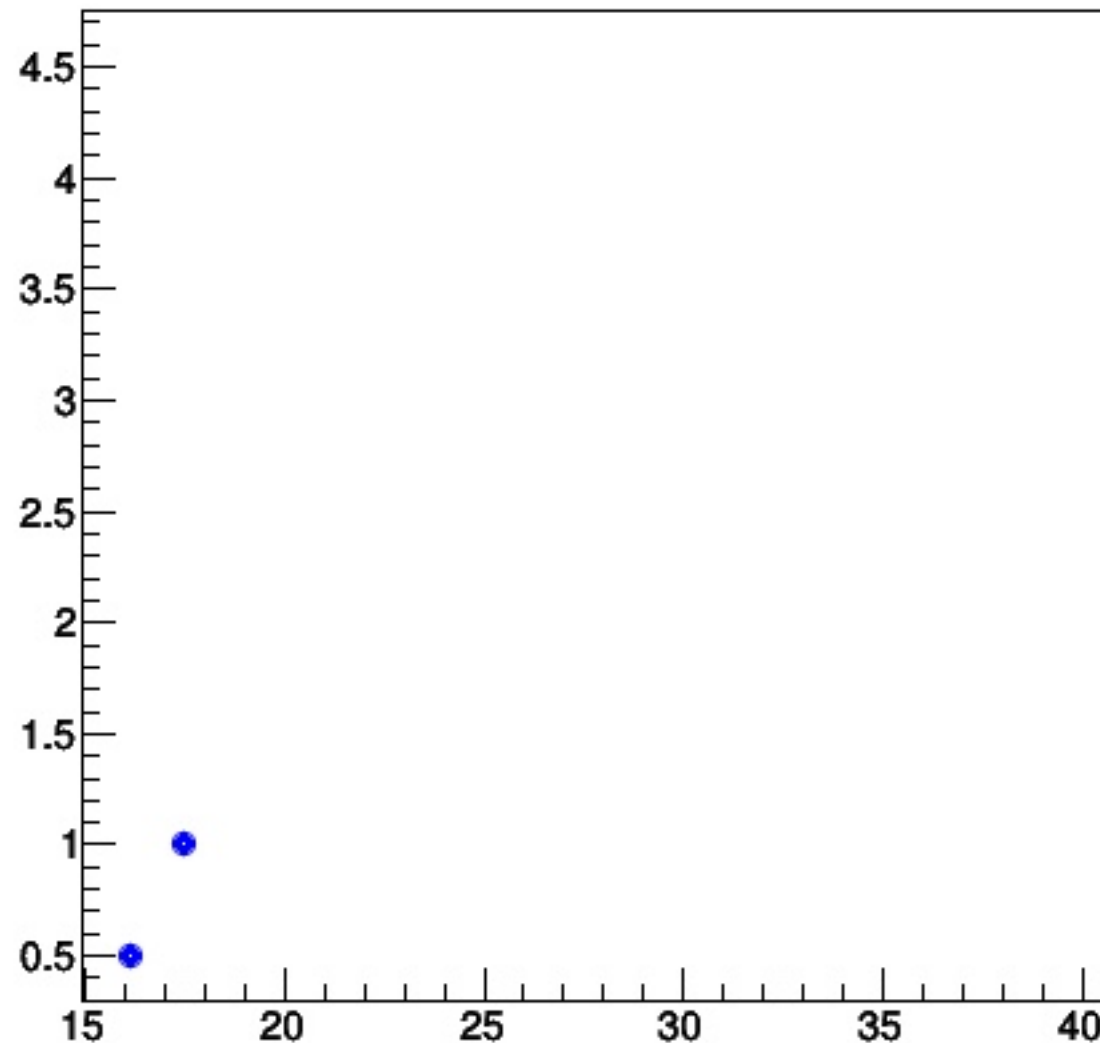
Hough Transform — Visualization Points

- Create lines going through hit point $(x,y)_i$
- **Repeat** for every hit point i



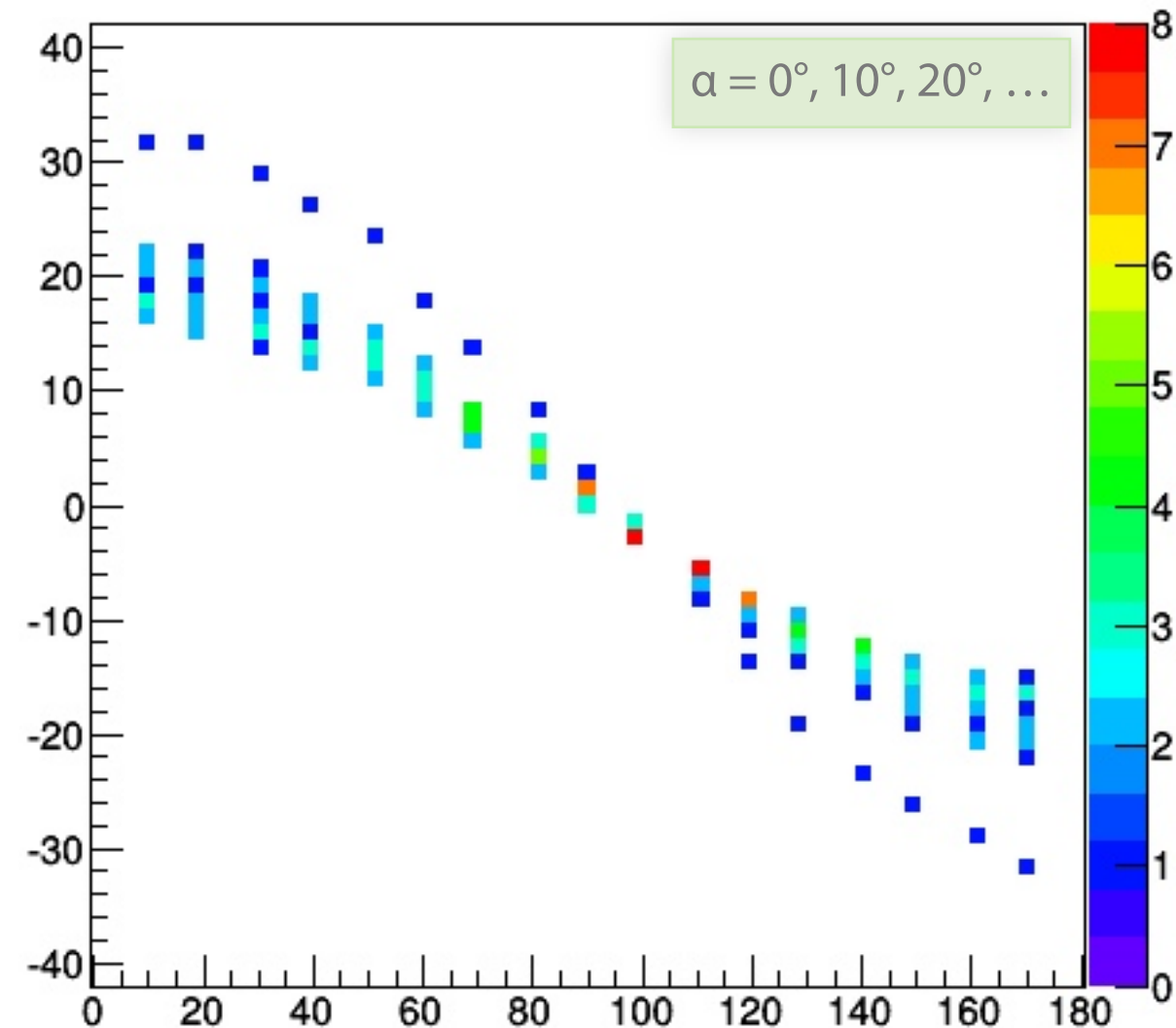
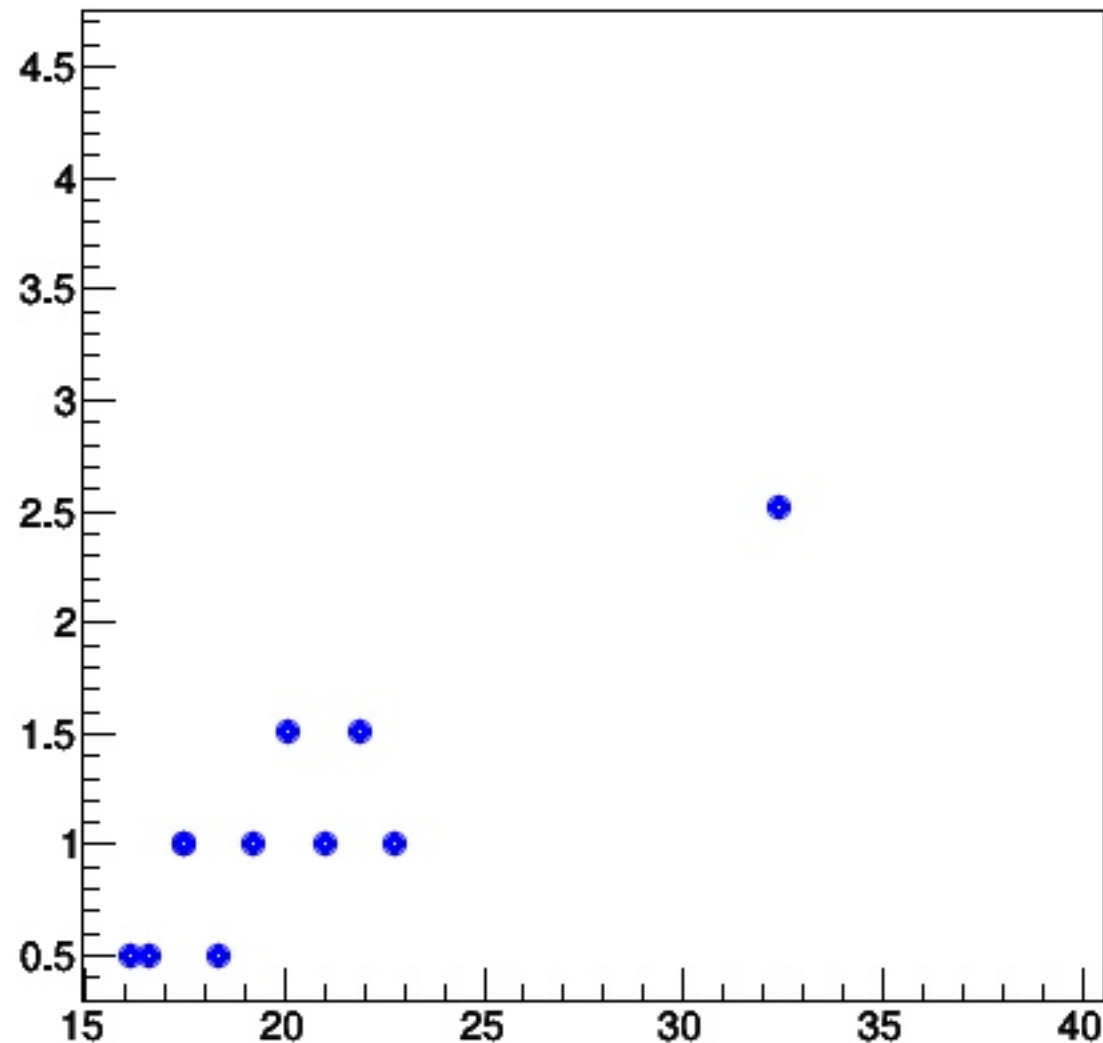
Hough Transform — Visualization Points

- Create lines going through hit point $(x,y)_i$
- **Repeat** for every hit point i



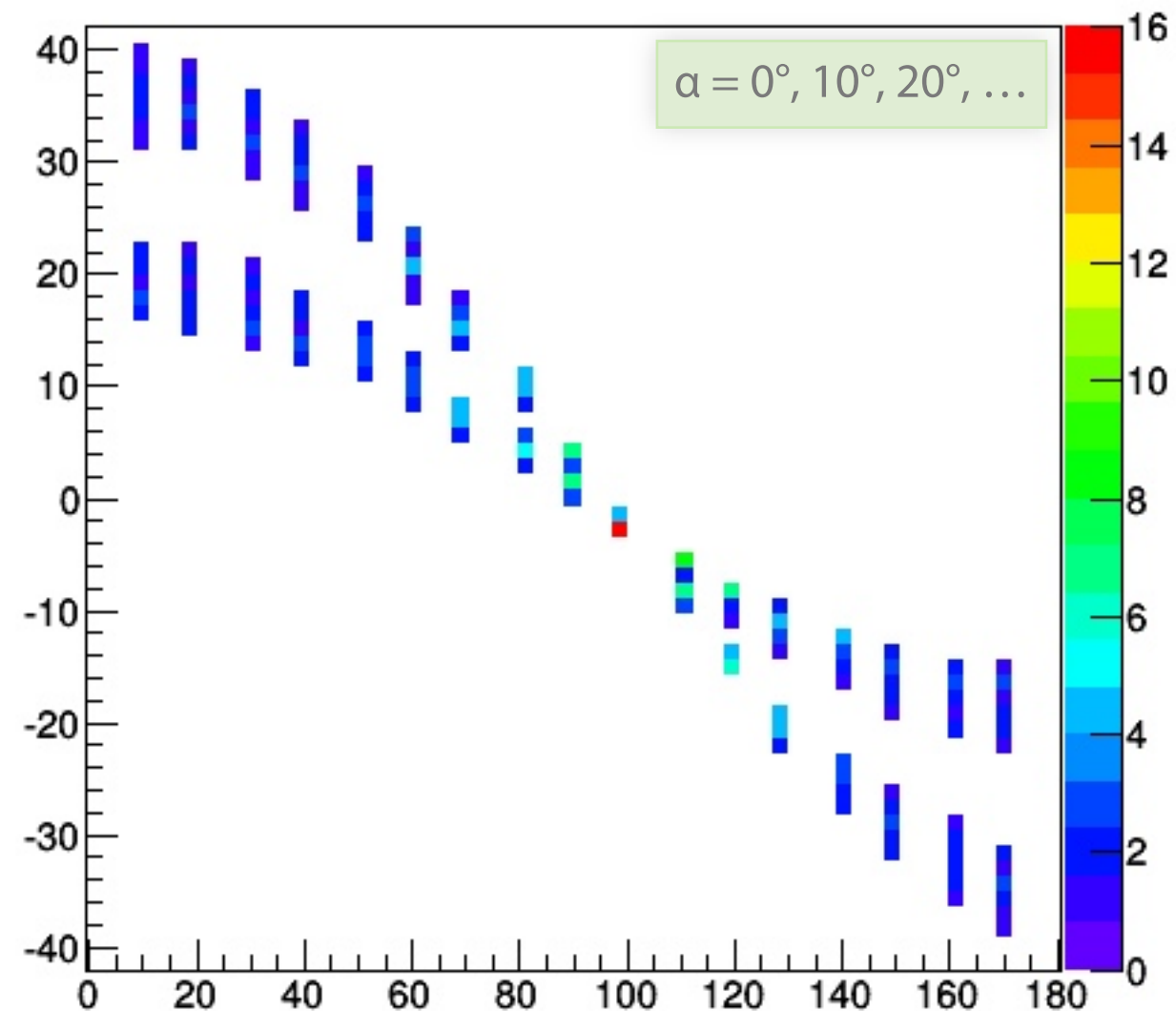
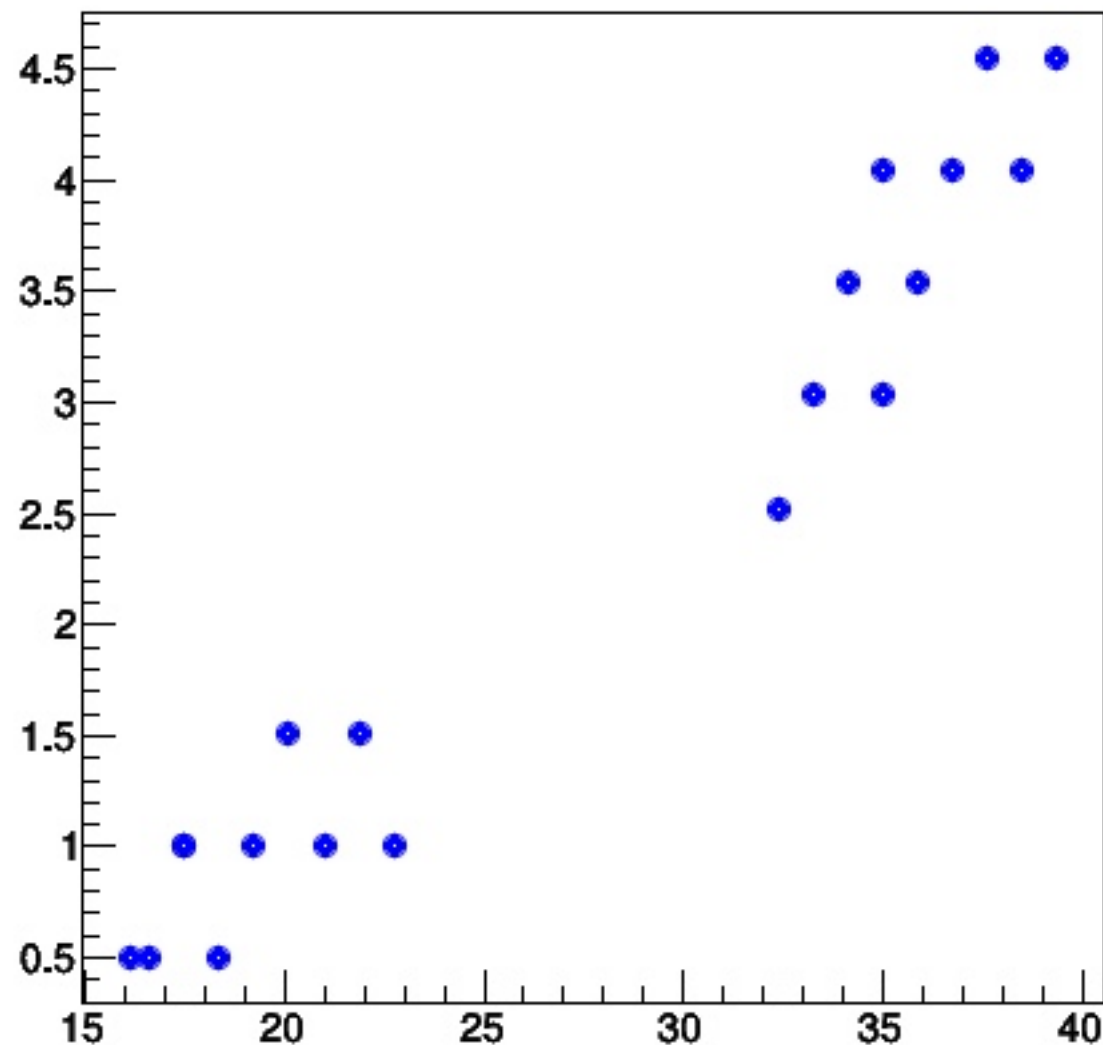
Hough Transform — Visualization Points

- Create lines going through hit point $(x,y)_i$
- **Repeat** for every hit point i



Hough Transform — Visualization Points

- Create lines going through hit point $(x,y)_i$
- **Repeat** for every hit point i



Hough Transform — Granularity

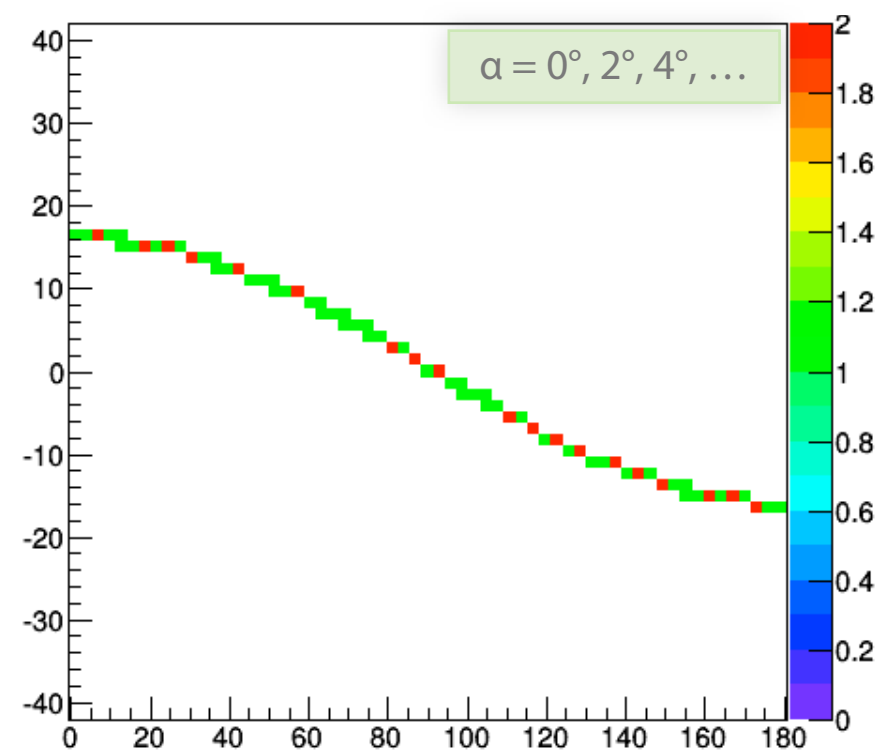
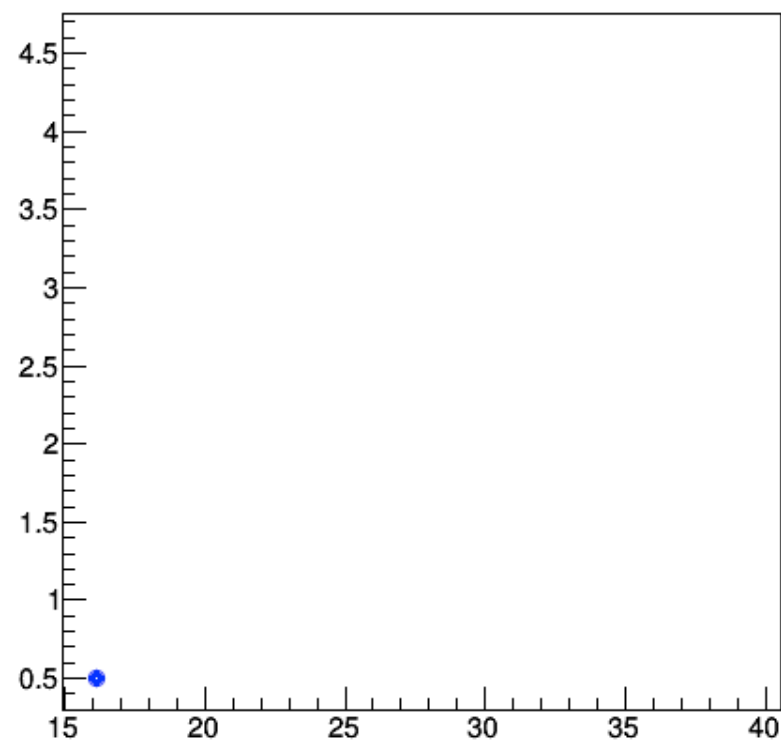
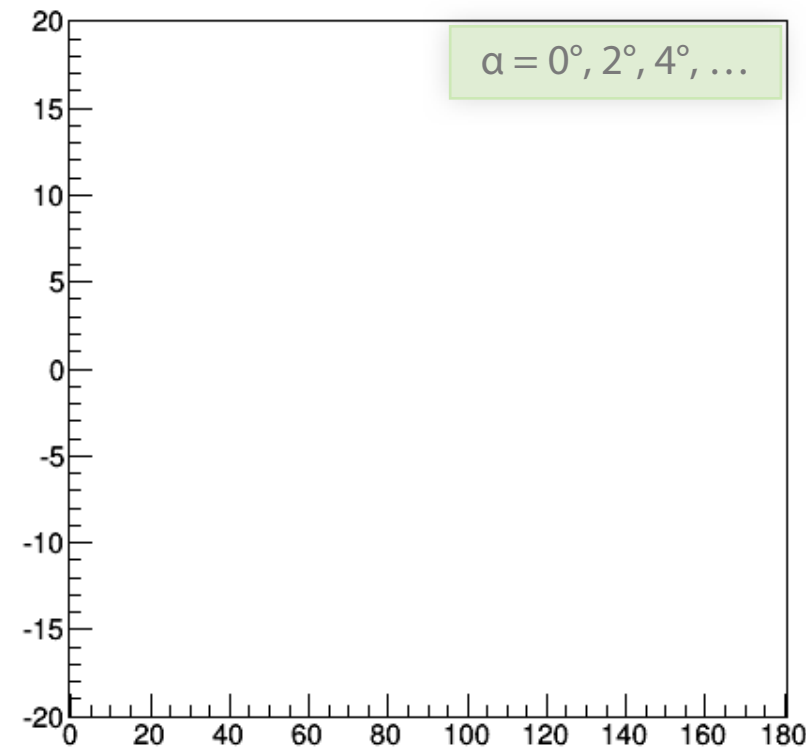
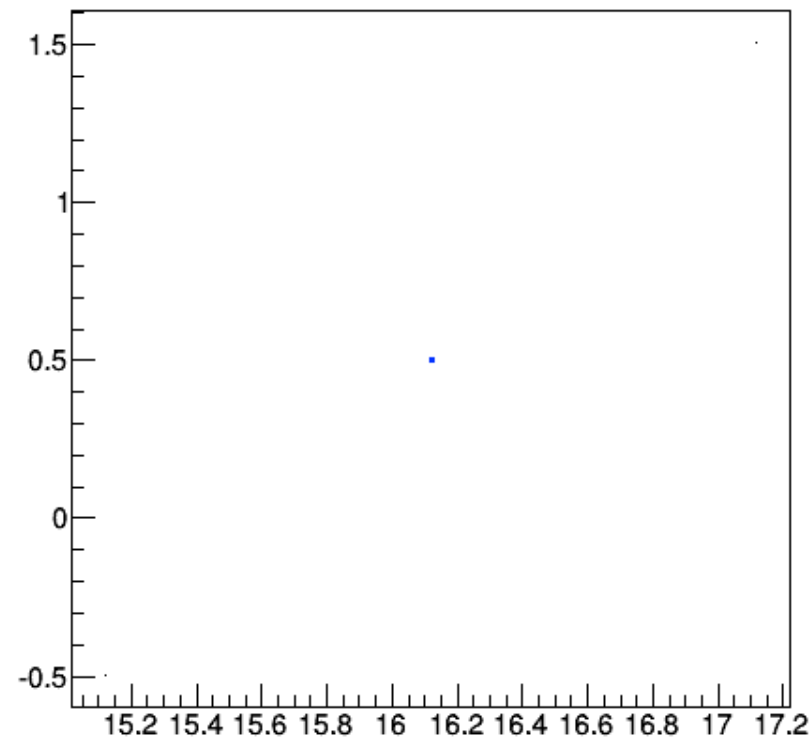
- Choice of α granularity determines resolution

$$\alpha = 0^\circ, 2^\circ, 4^\circ, \dots$$

$$\alpha = 0^\circ, 2^\circ, 4^\circ, \dots$$

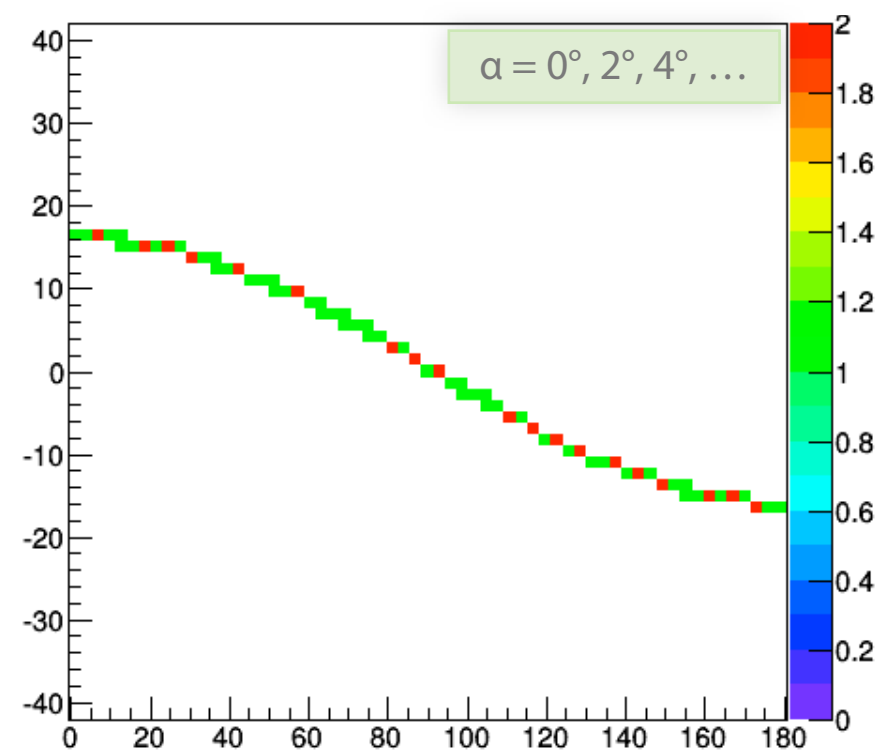
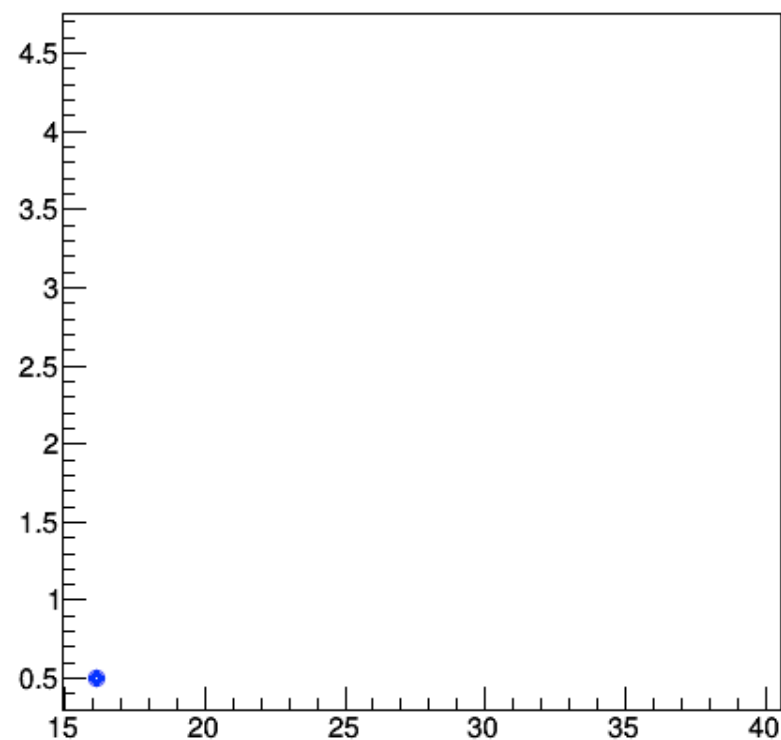
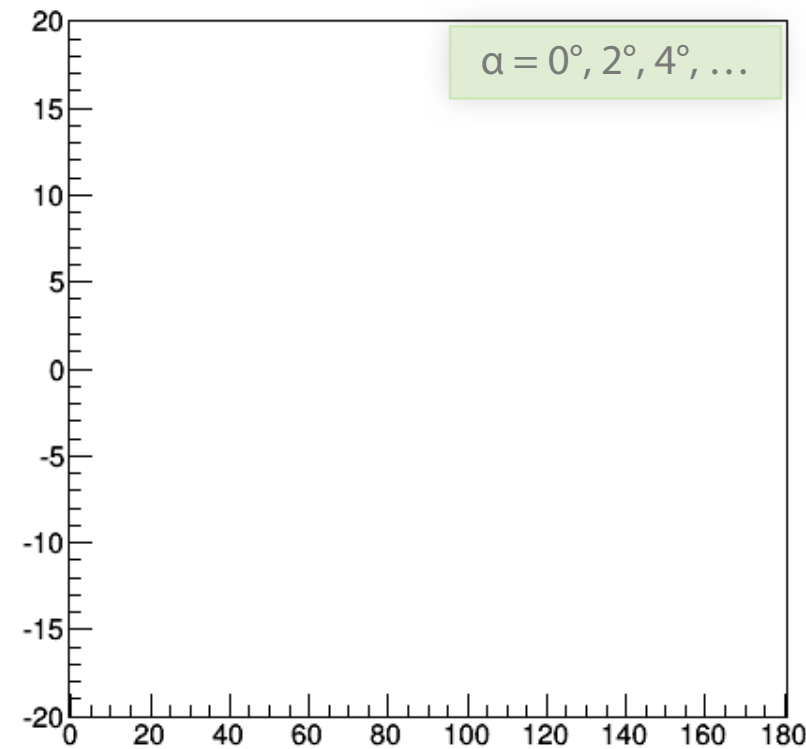
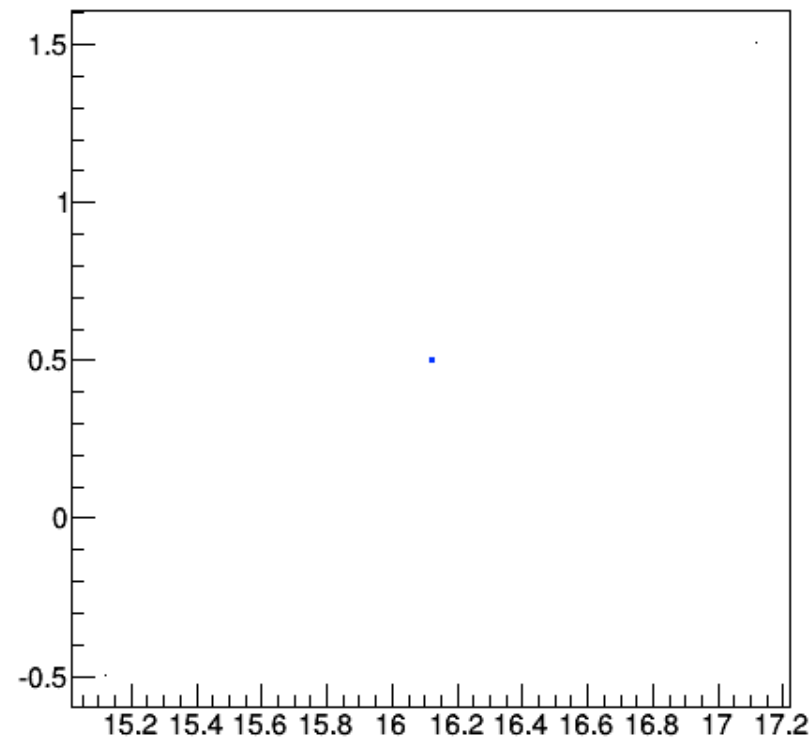
Hough Transform — Granularity

- Choice of α granularity determines resolution



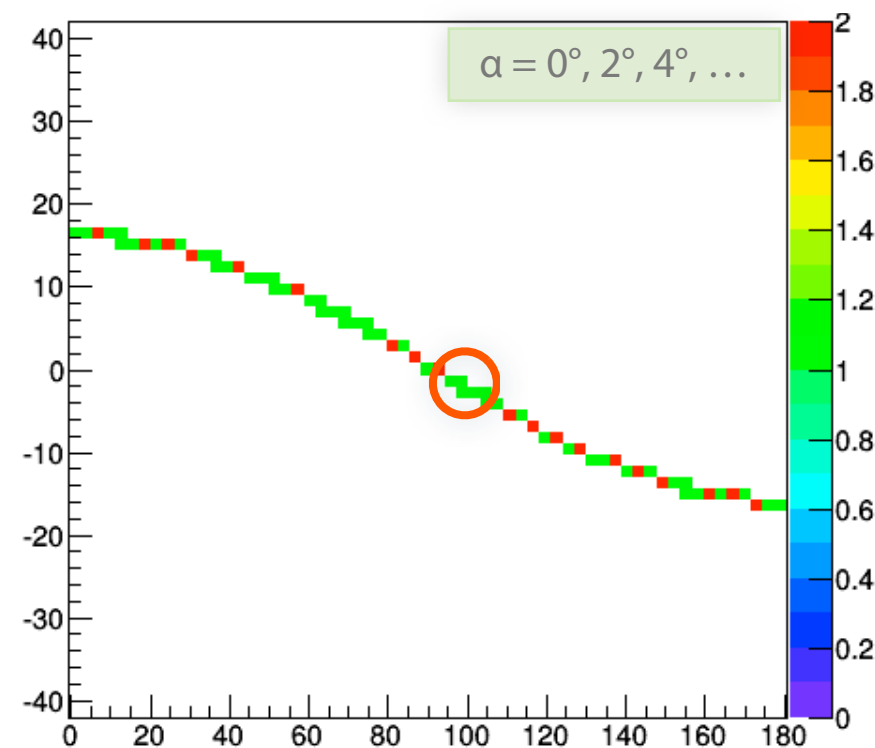
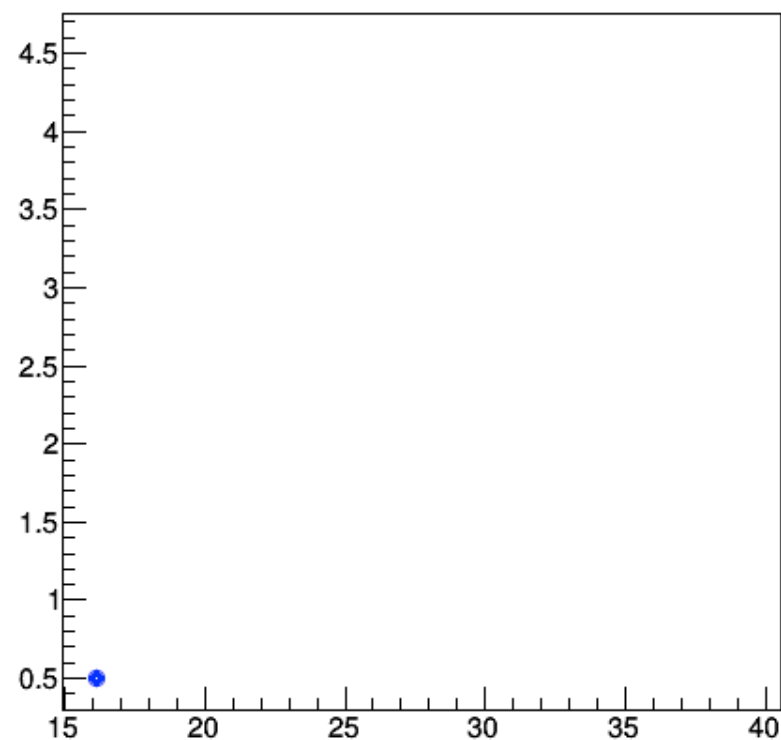
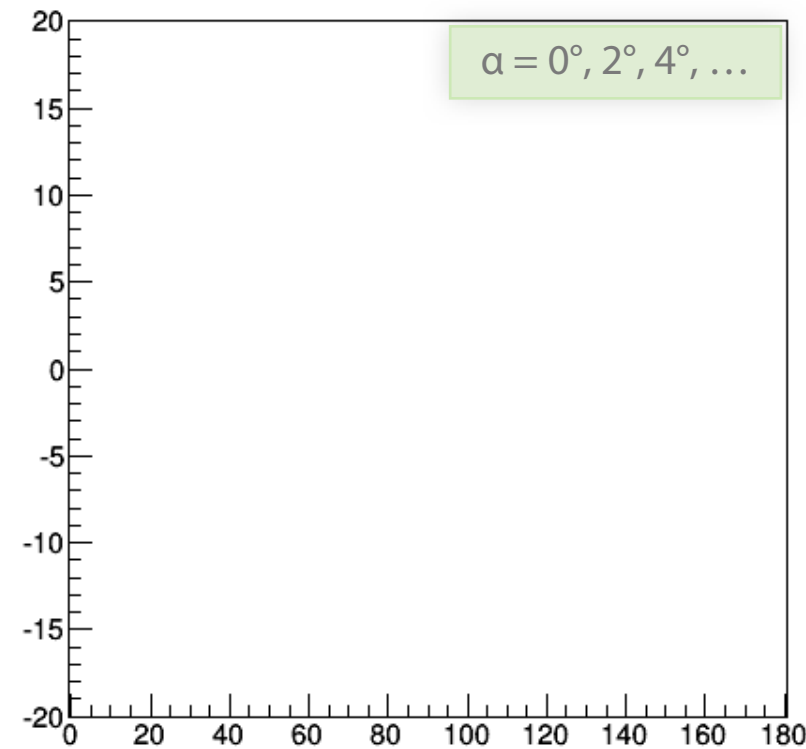
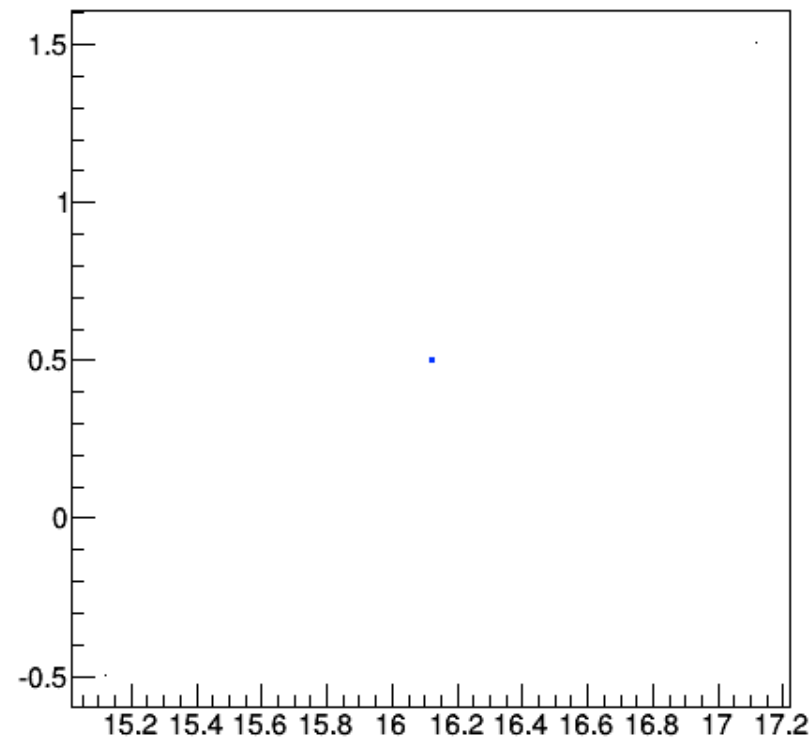
Hough Transform — Granularity

- Choice of α granularity determines resolution



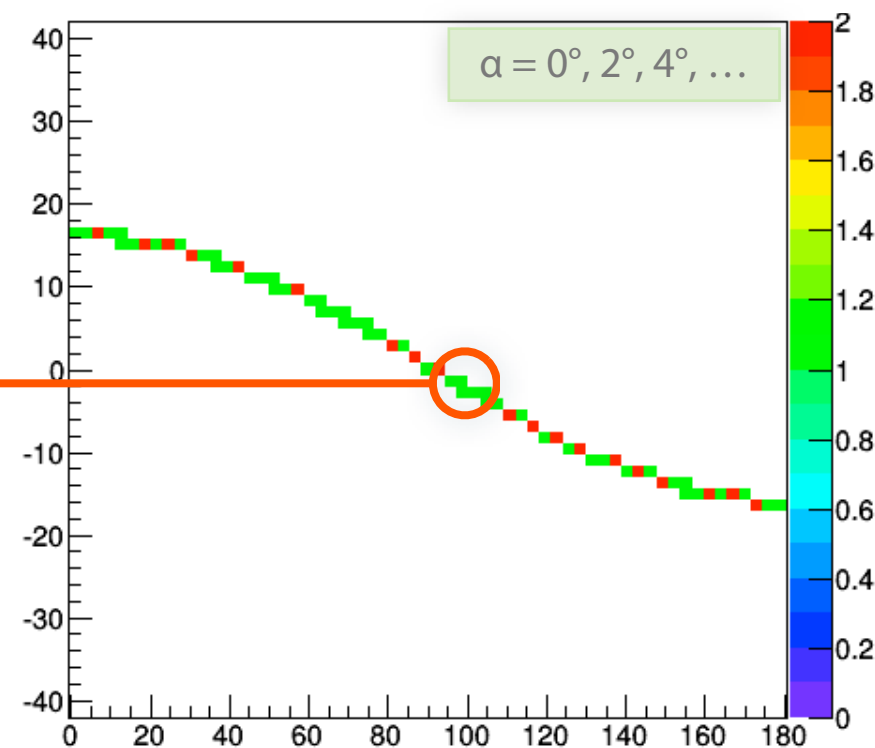
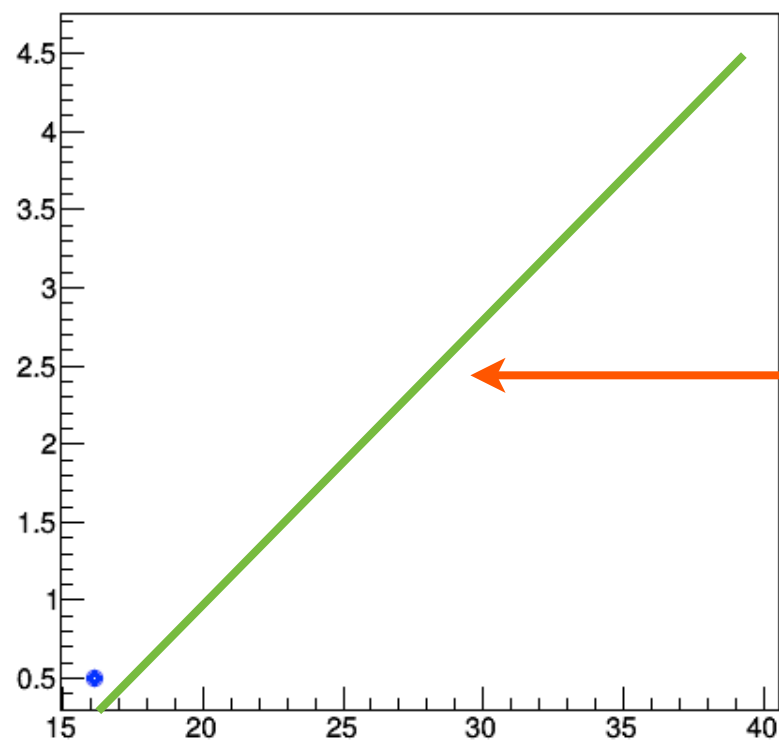
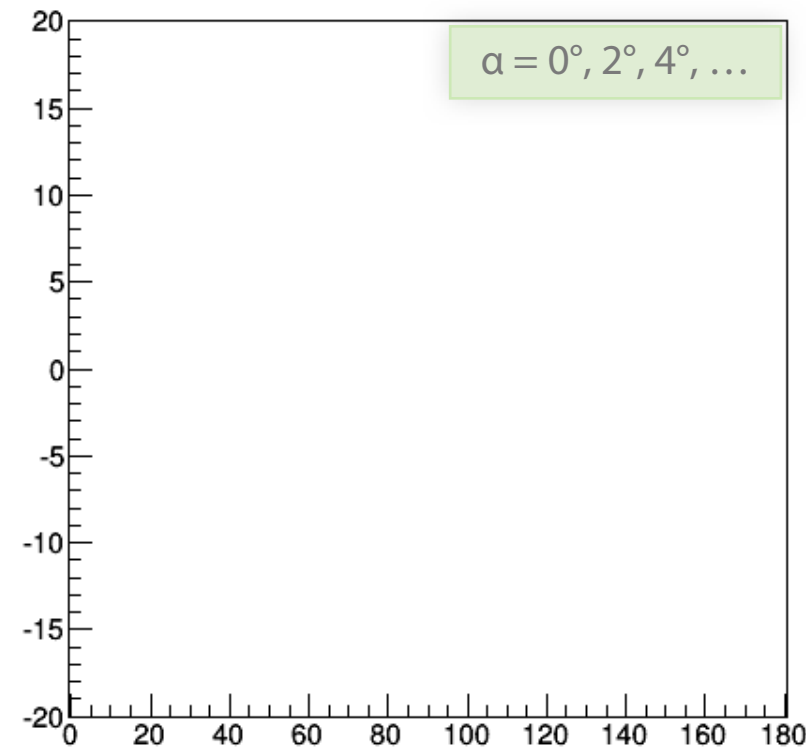
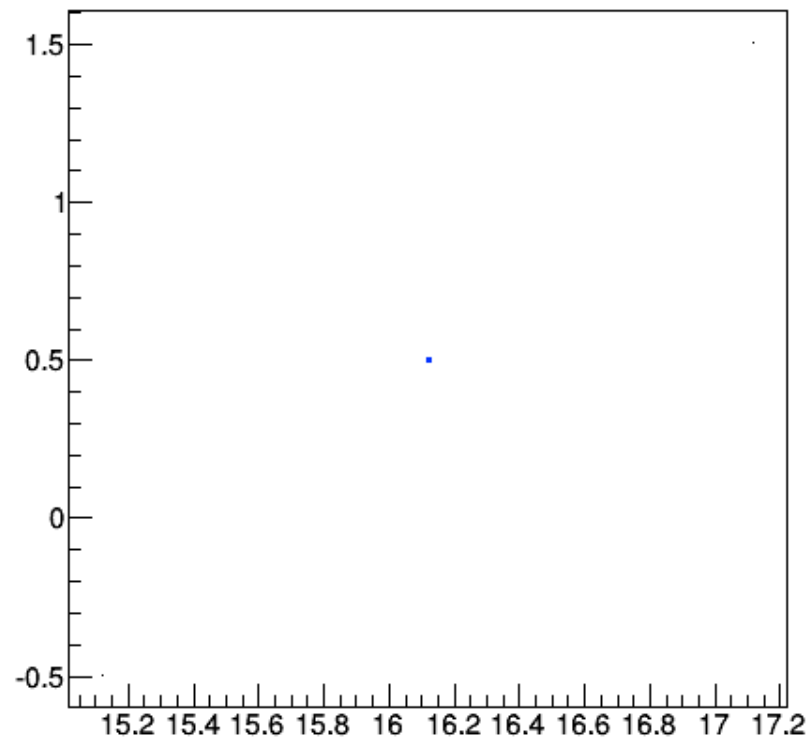
Hough Transform — Granularity

- Choice of α granularity determines resolution



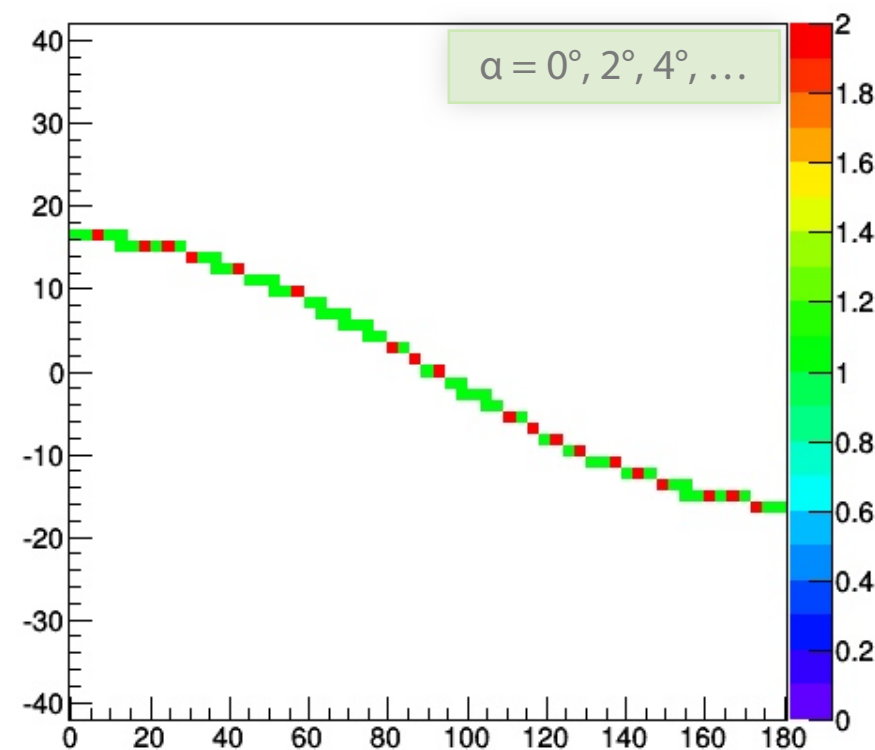
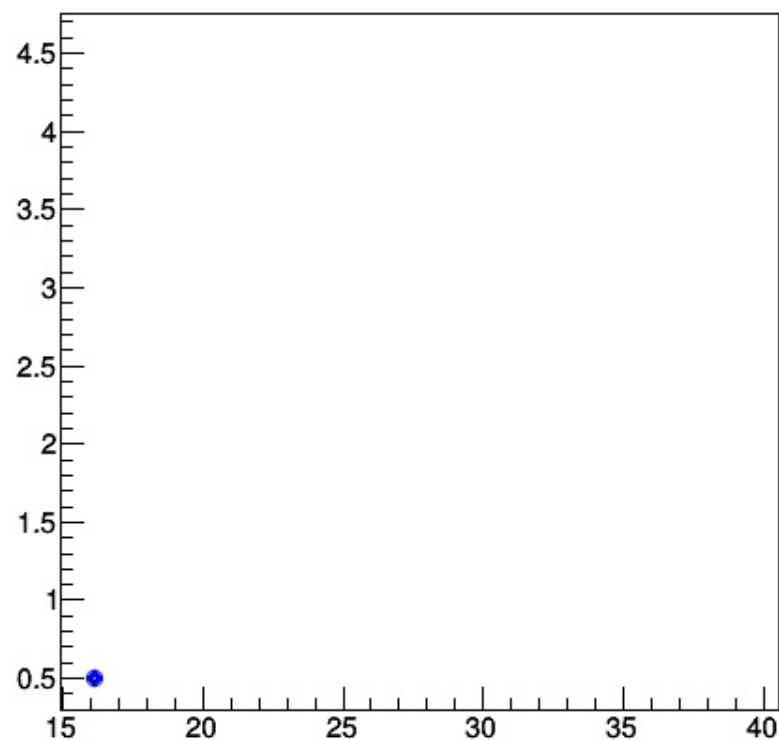
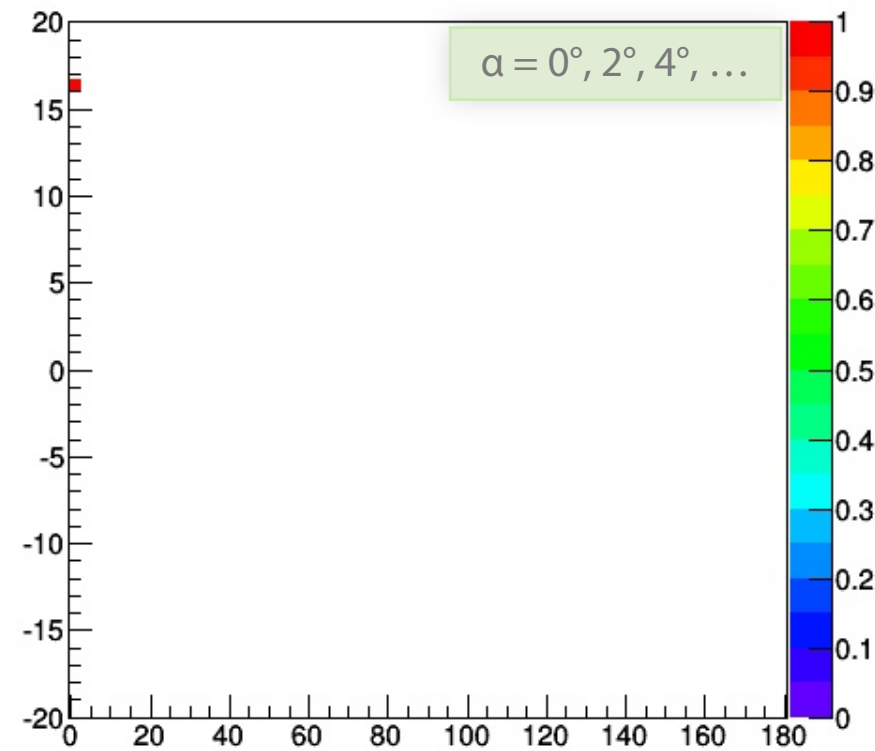
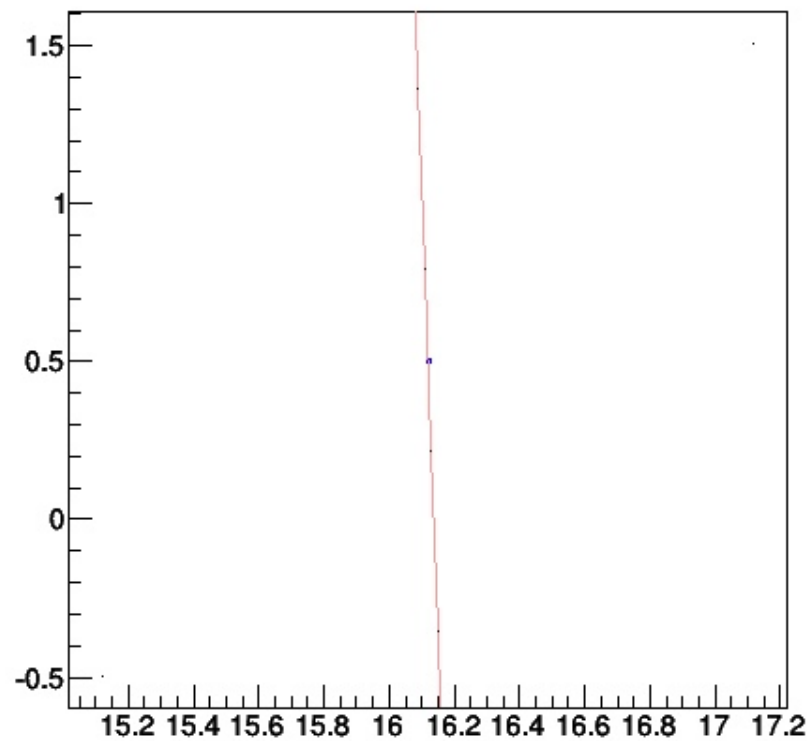
Hough Transform — Granularity

- Choice of α granularity determines resolution



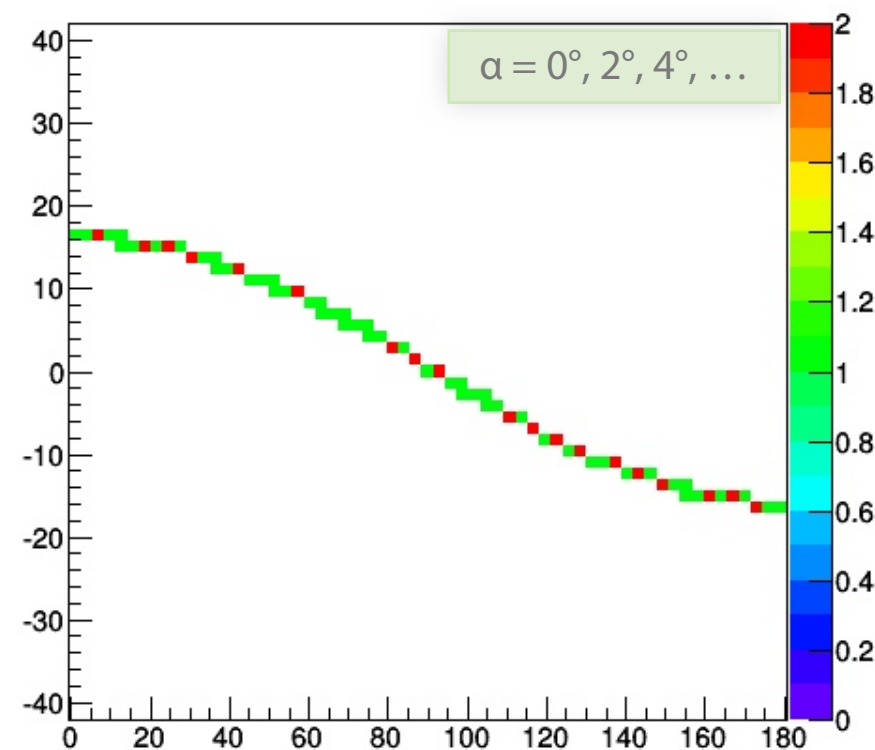
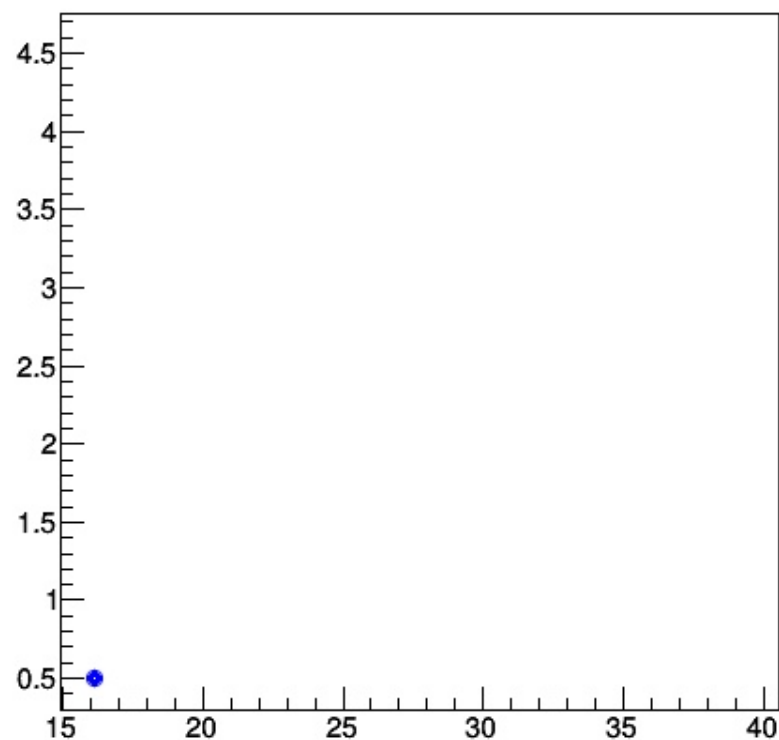
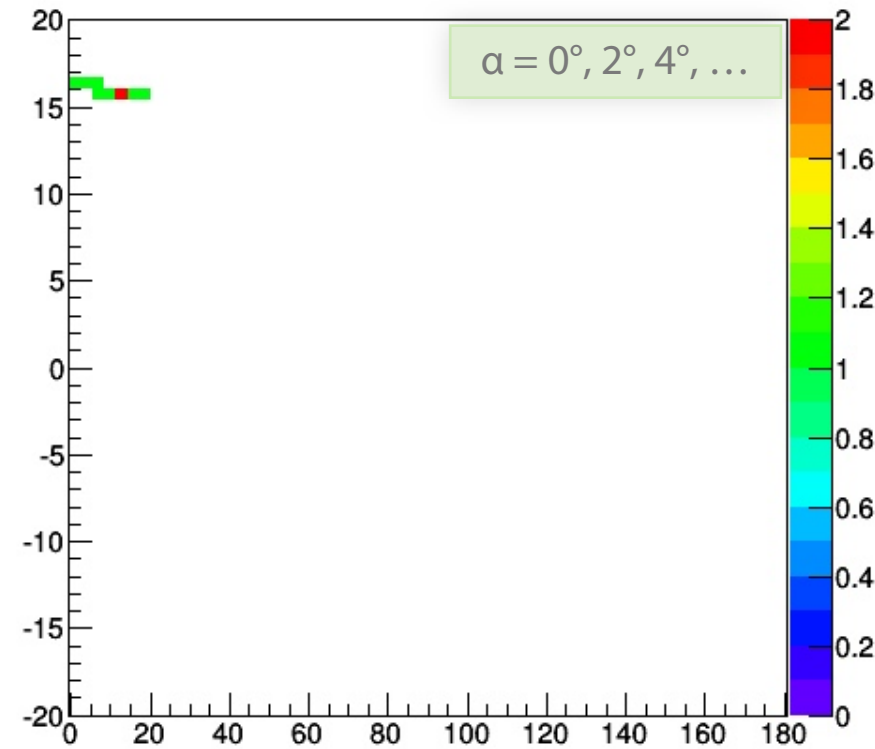
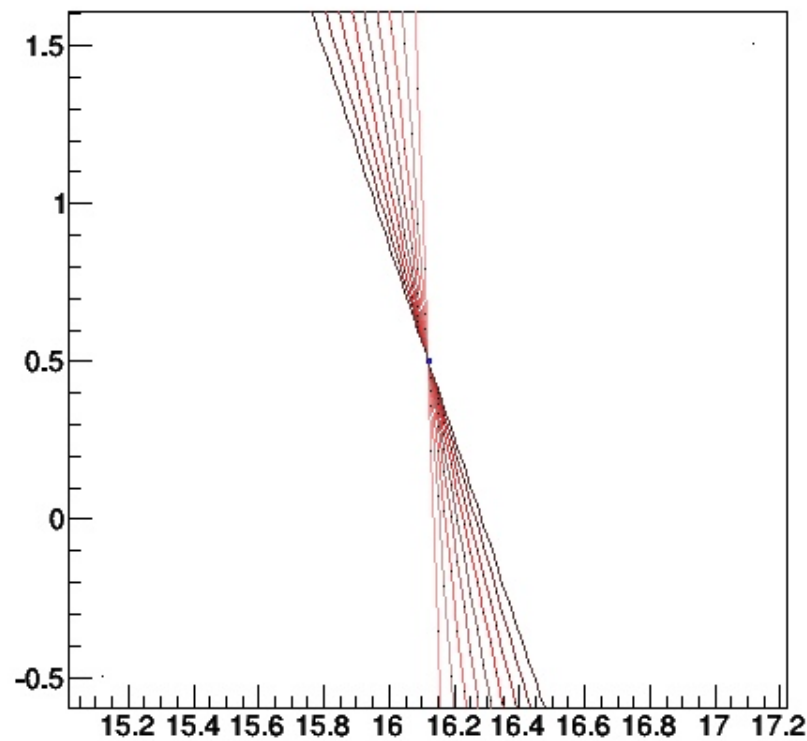
Hough Transform — Granularity

- Choice of α granularity determines resolution



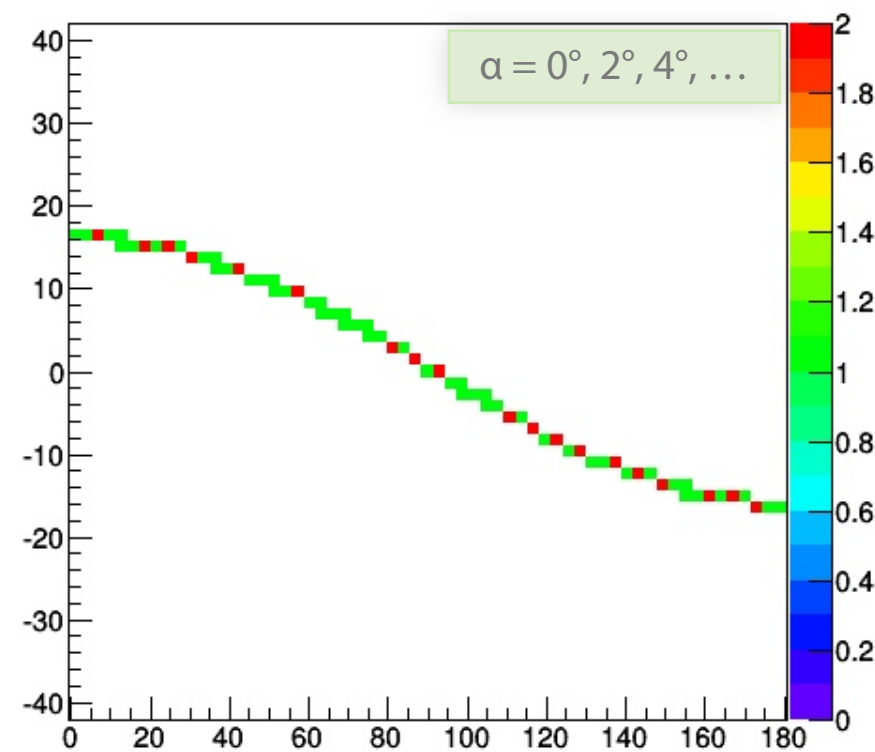
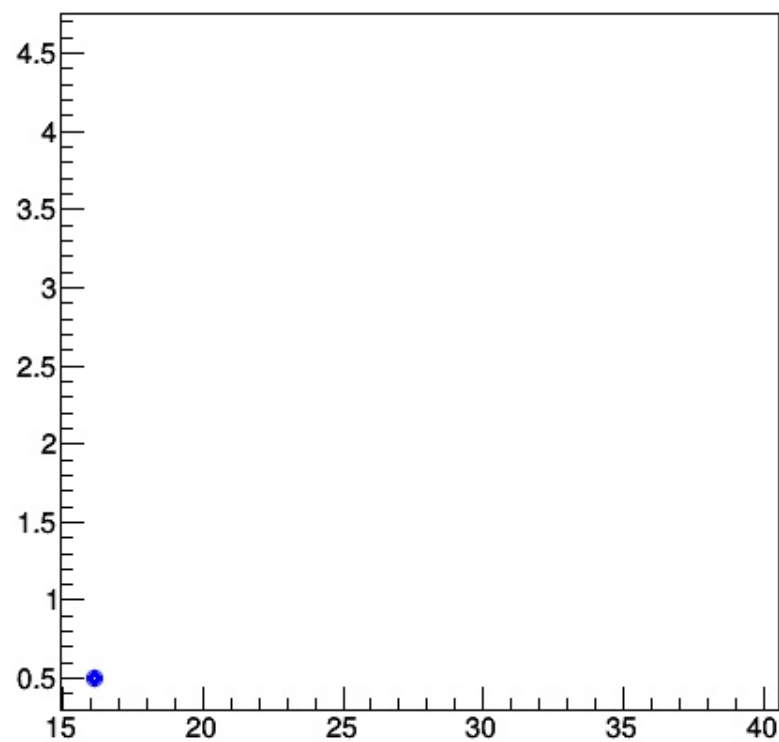
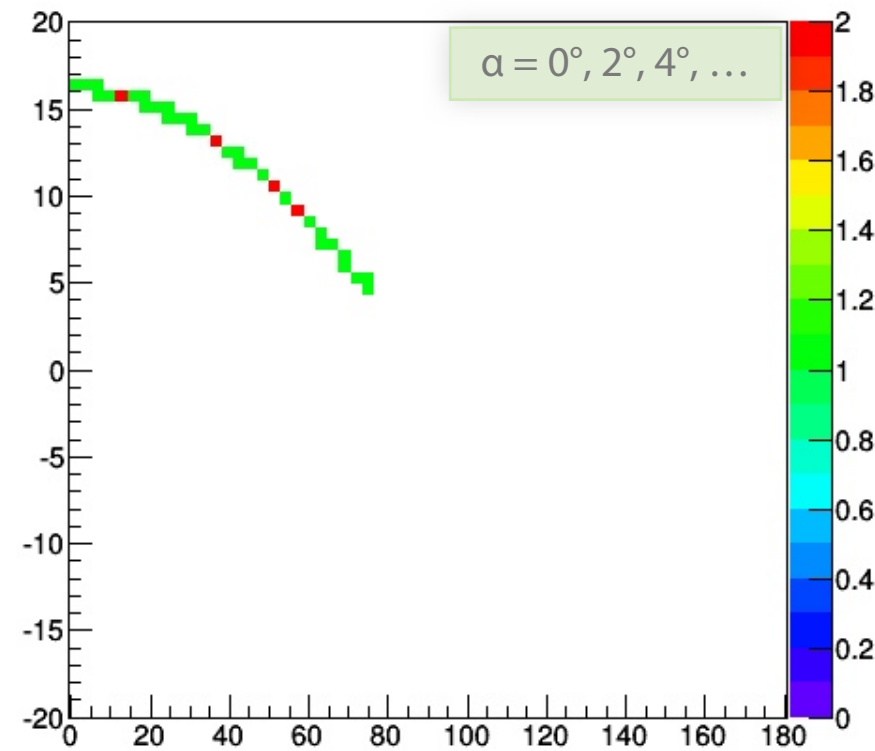
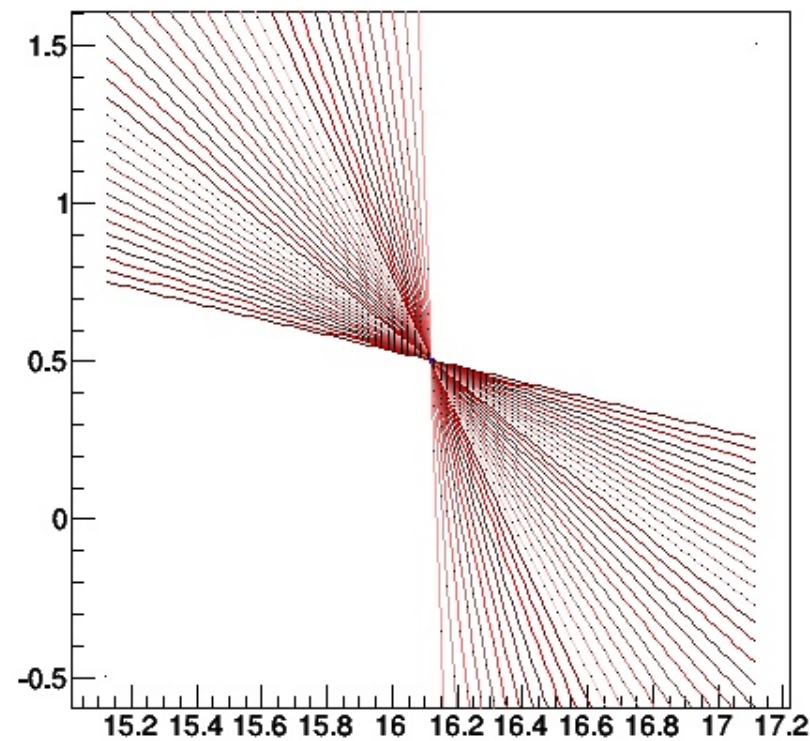
Hough Transform — Granularity

- Choice of α granularity determines resolution



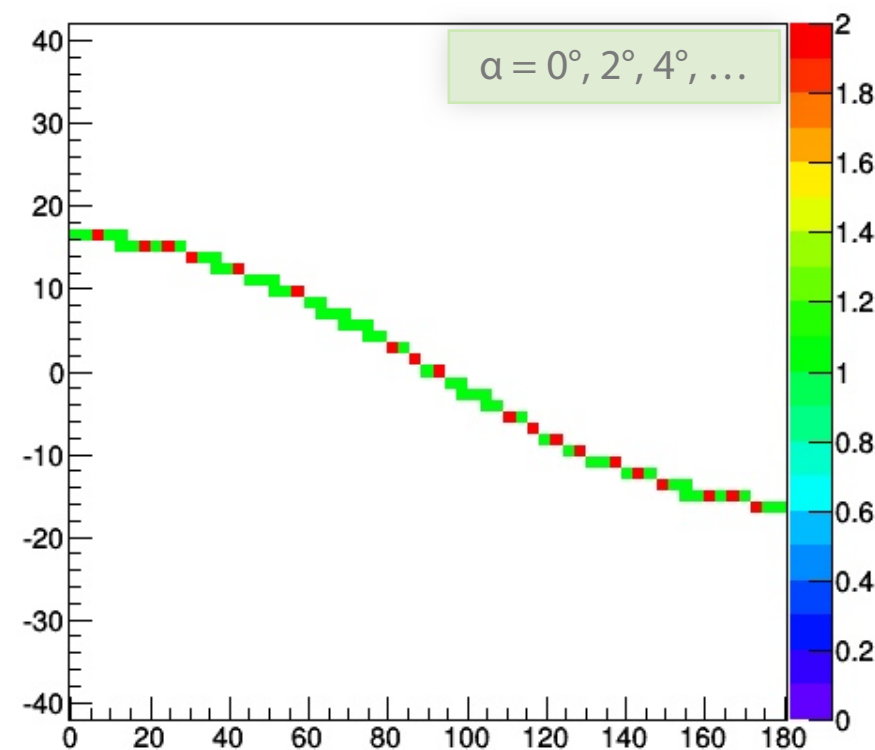
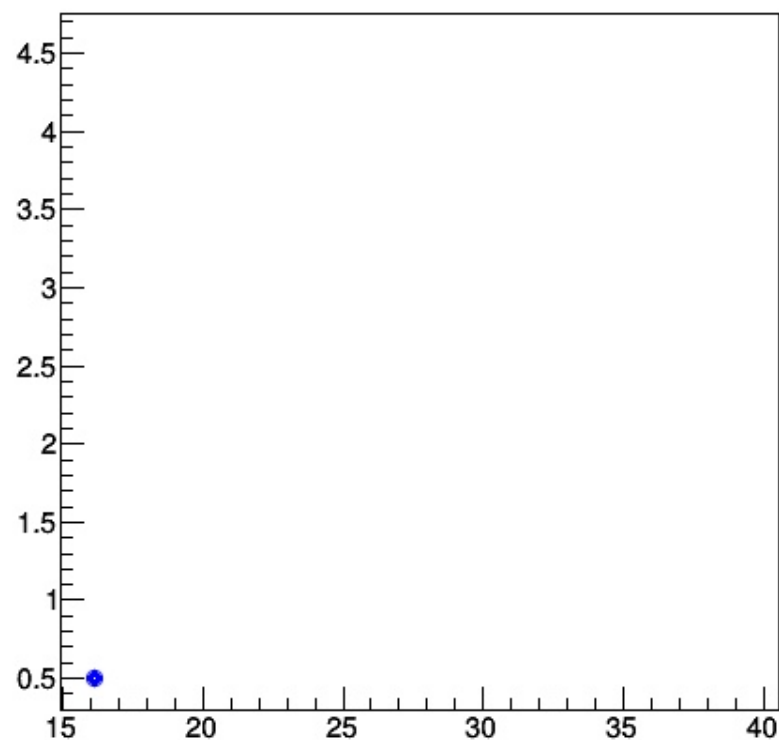
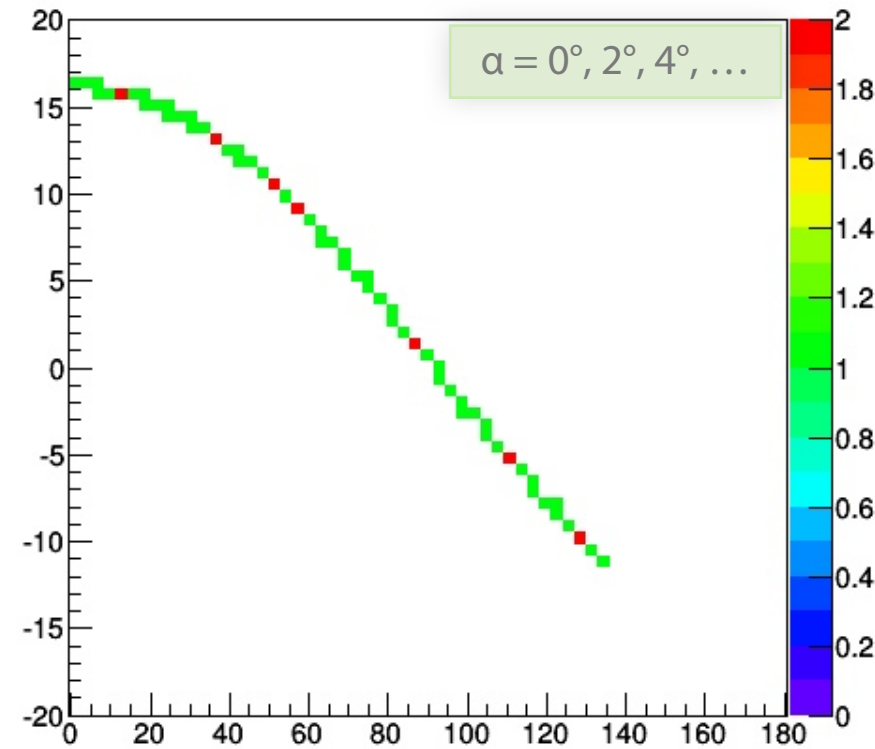
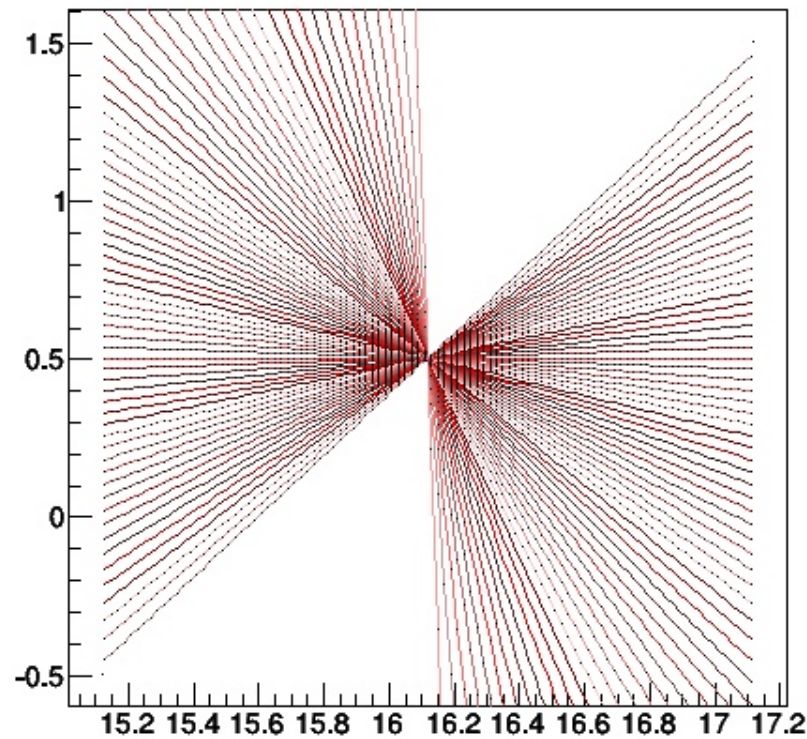
Hough Transform — Granularity

- Choice of α granularity determines resolution



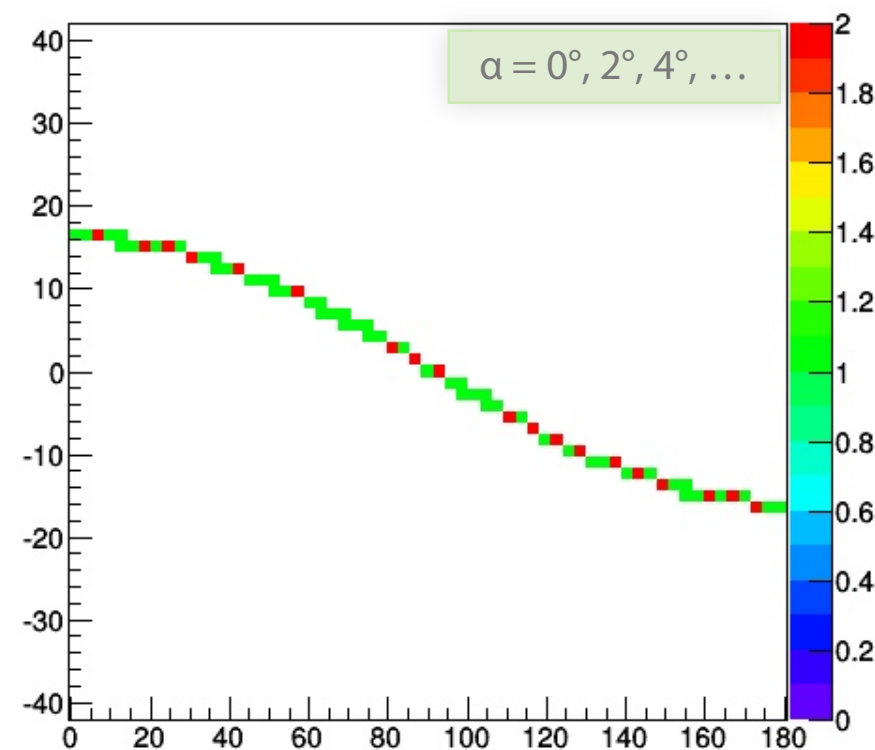
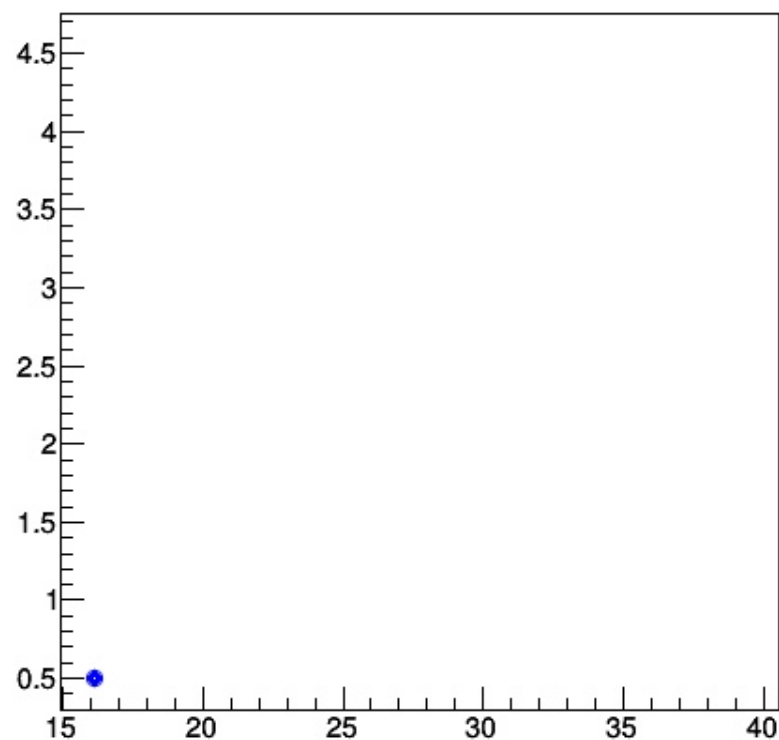
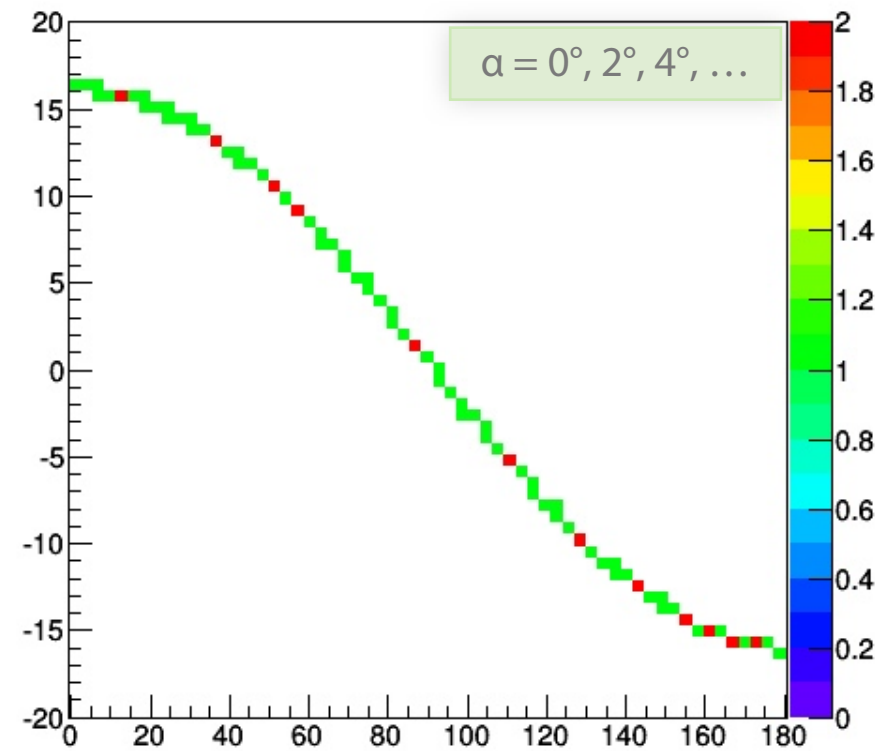
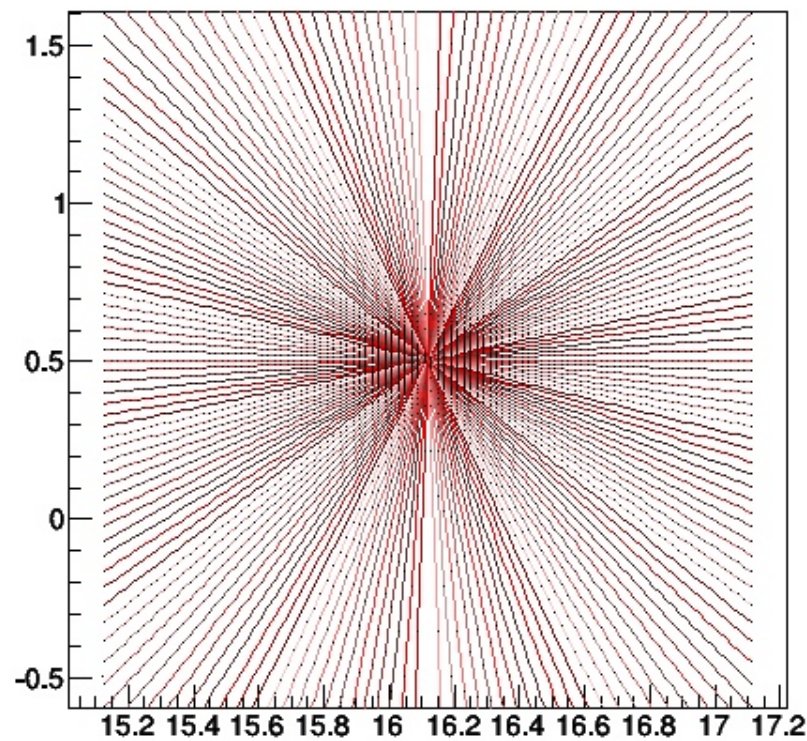
Hough Transform — Granularity

- Choice of α granularity determines resolution



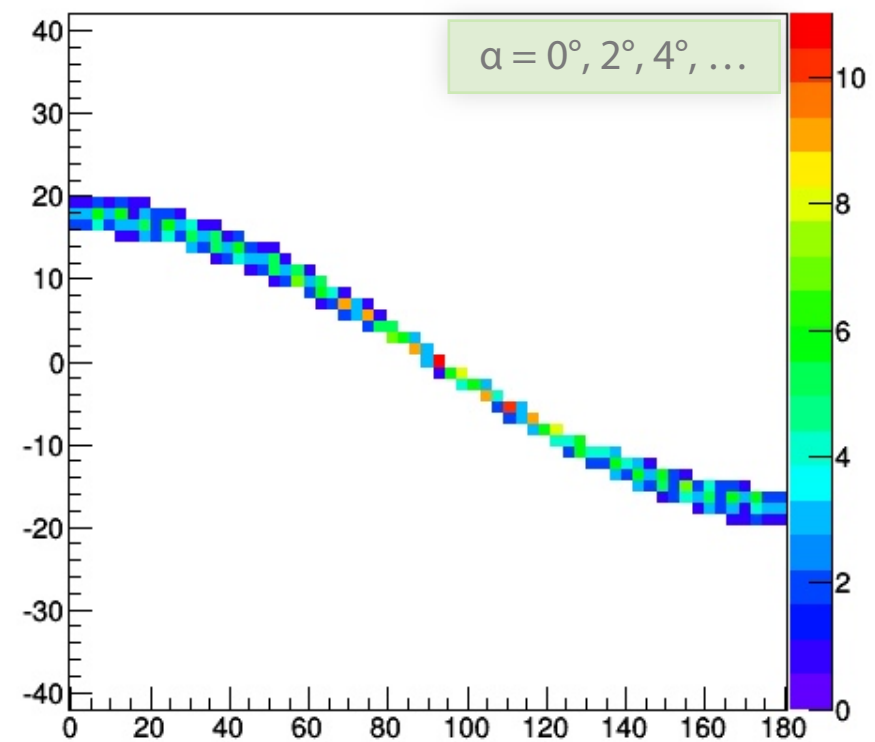
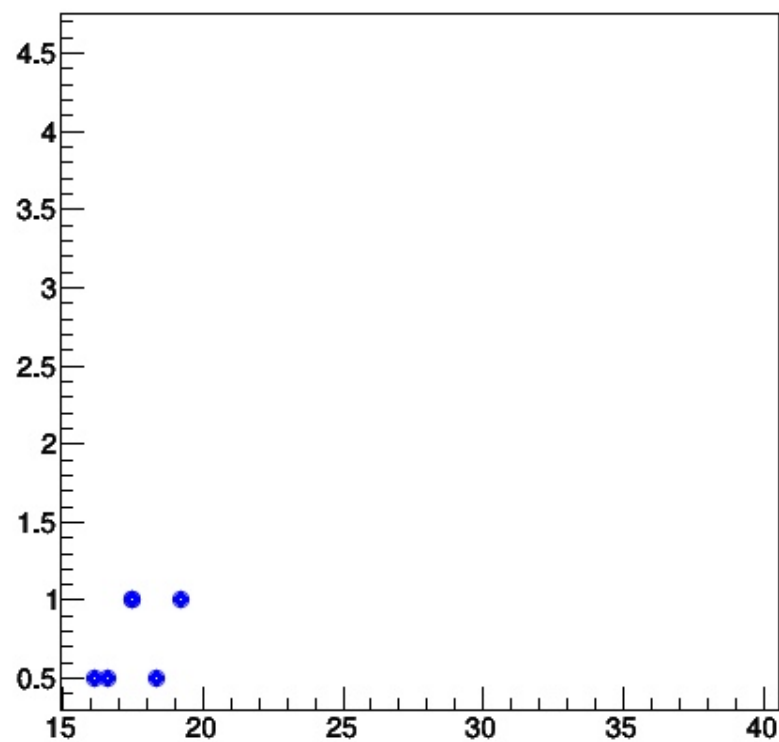
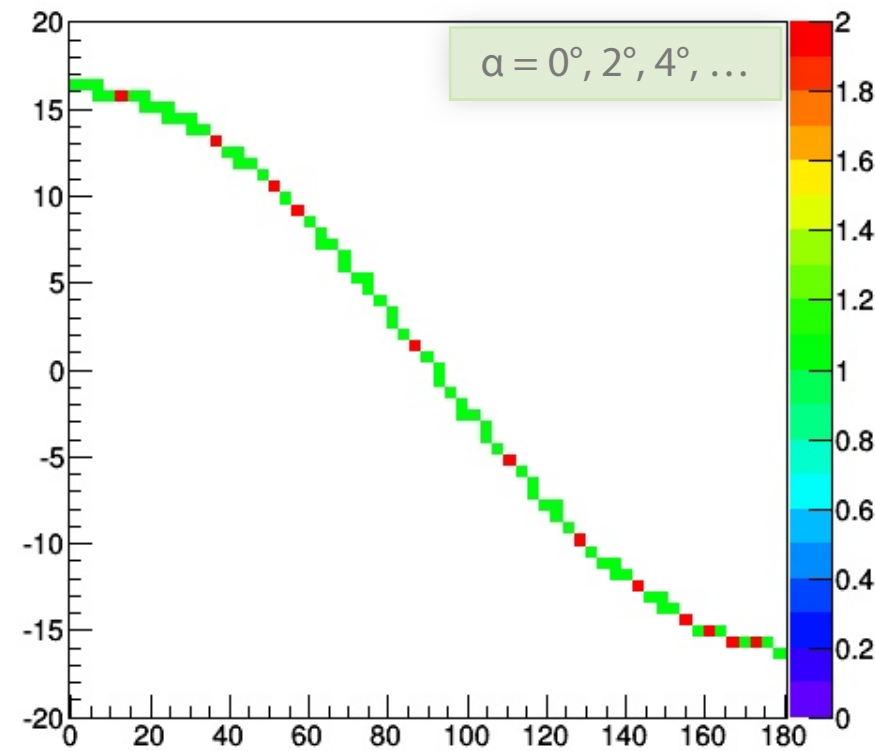
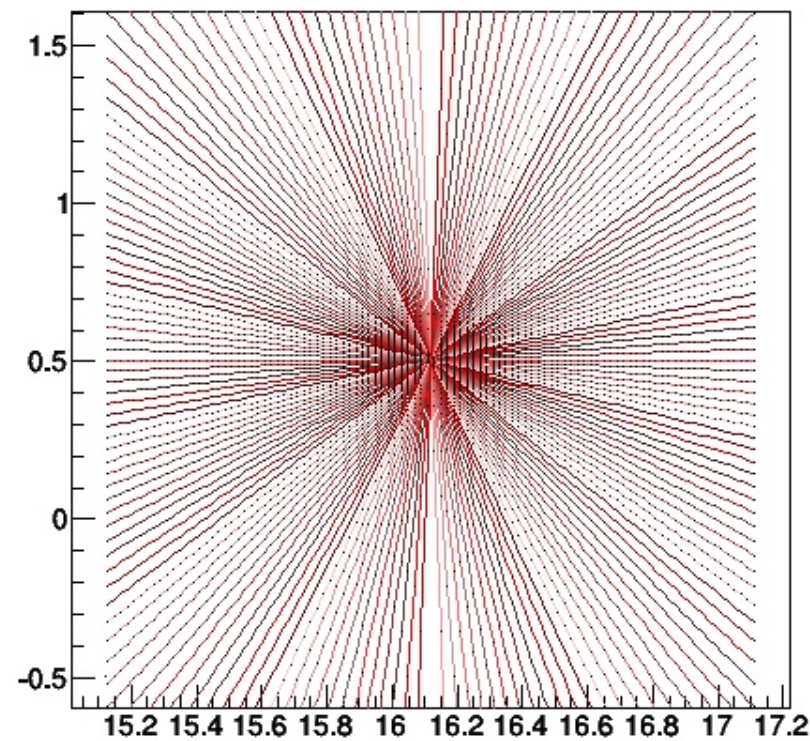
Hough Transform — Granularity

- Choice of α granularity determines resolution



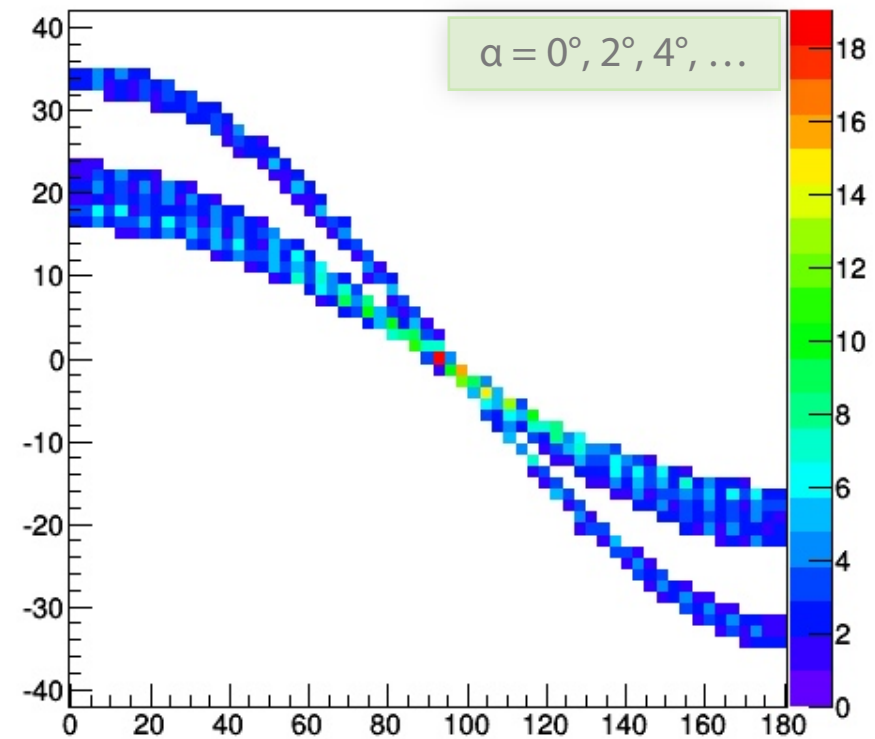
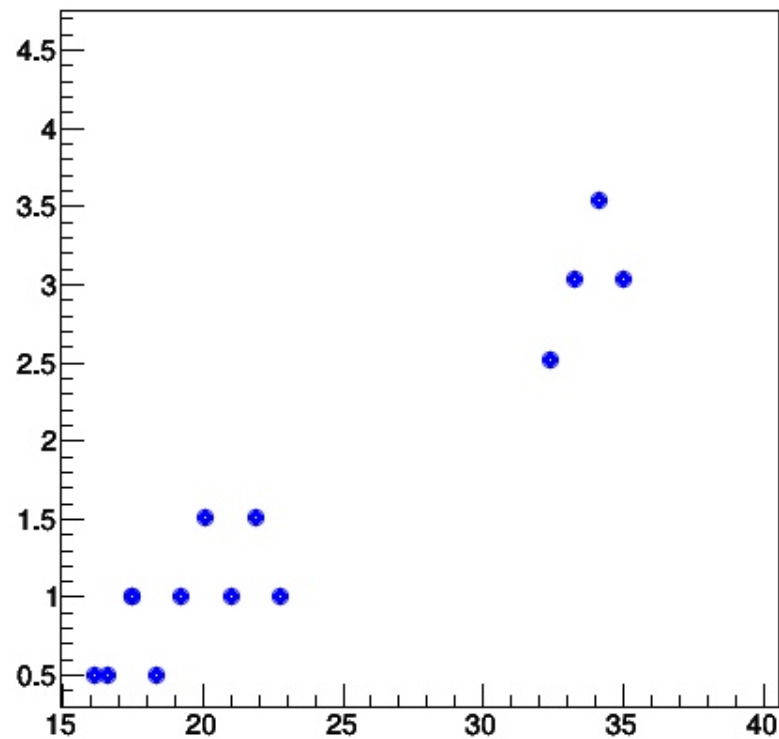
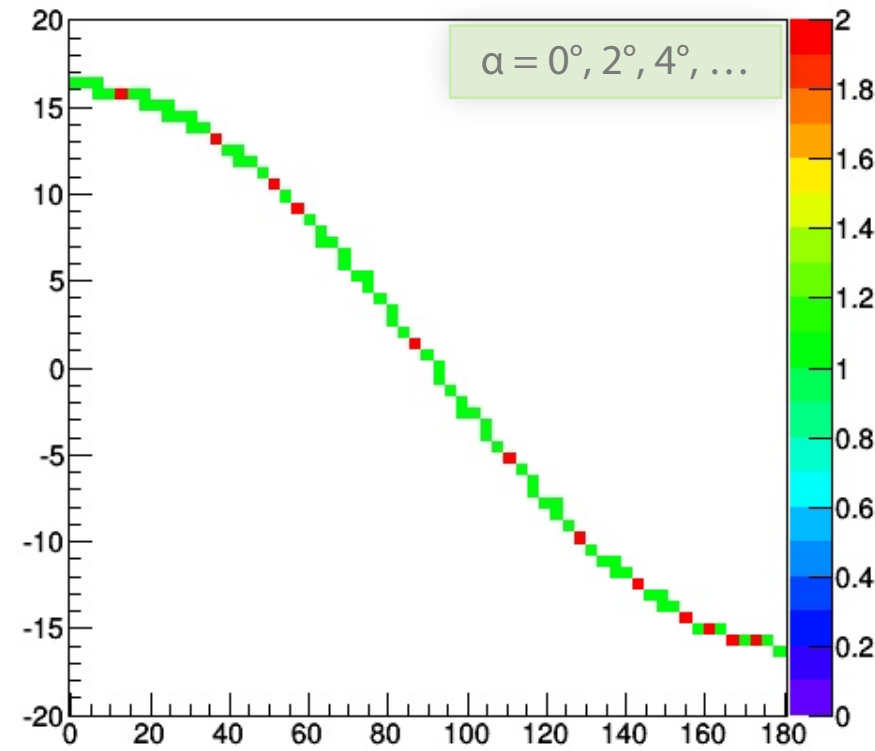
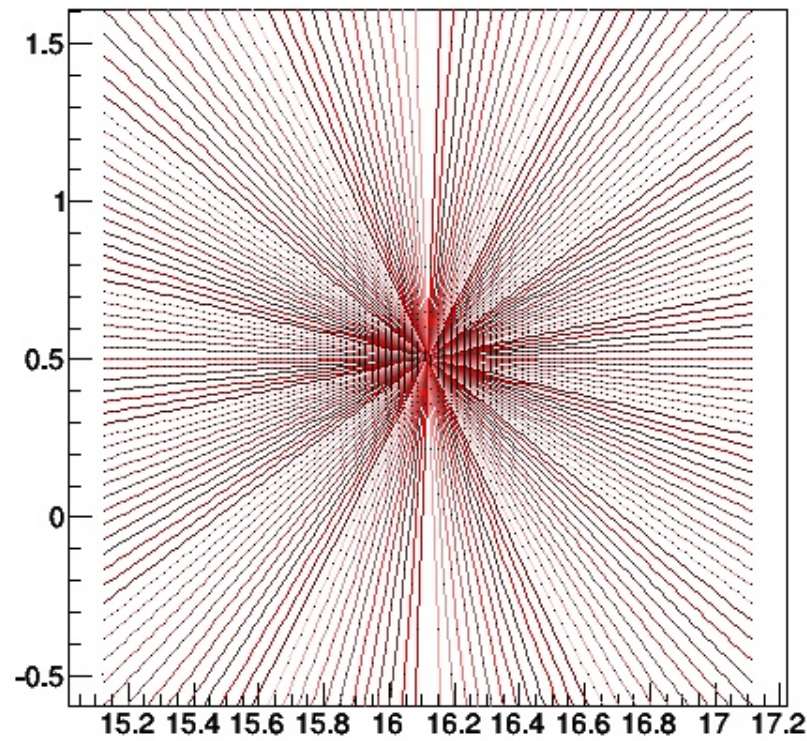
Hough Transform — Granularity

- Choice of α granularity determines resolution



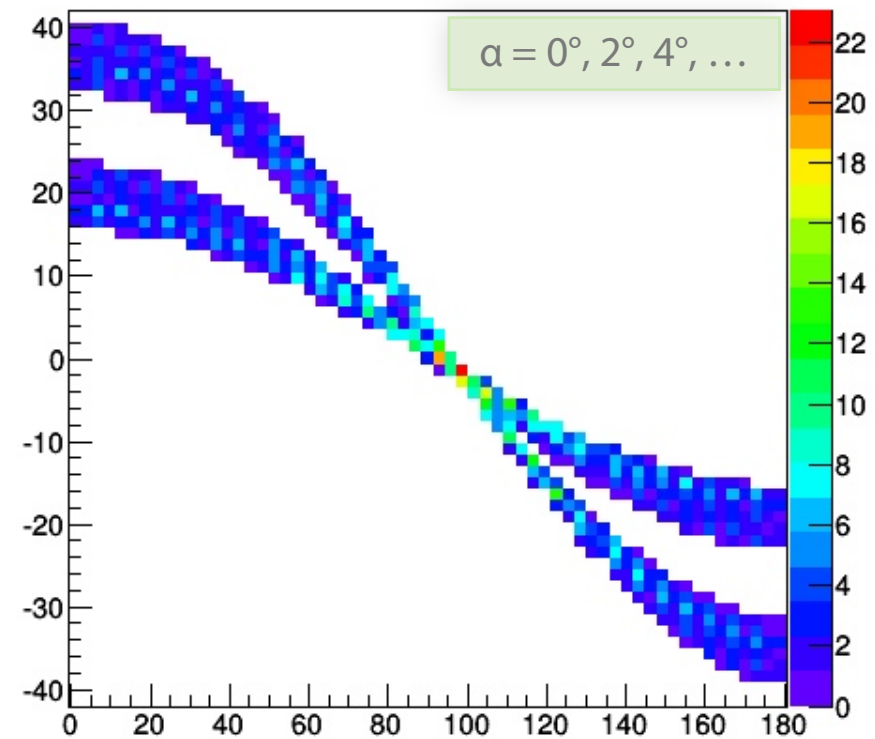
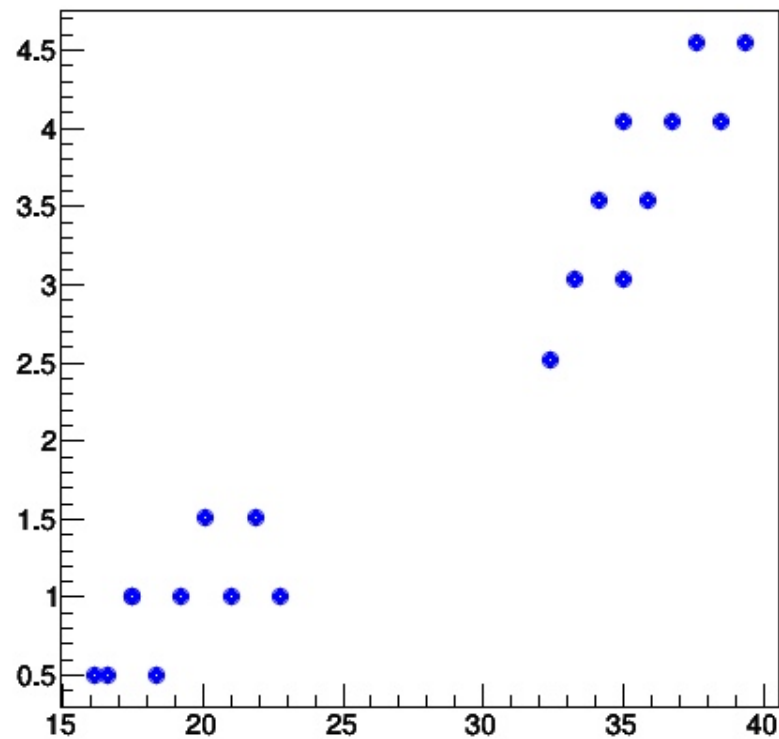
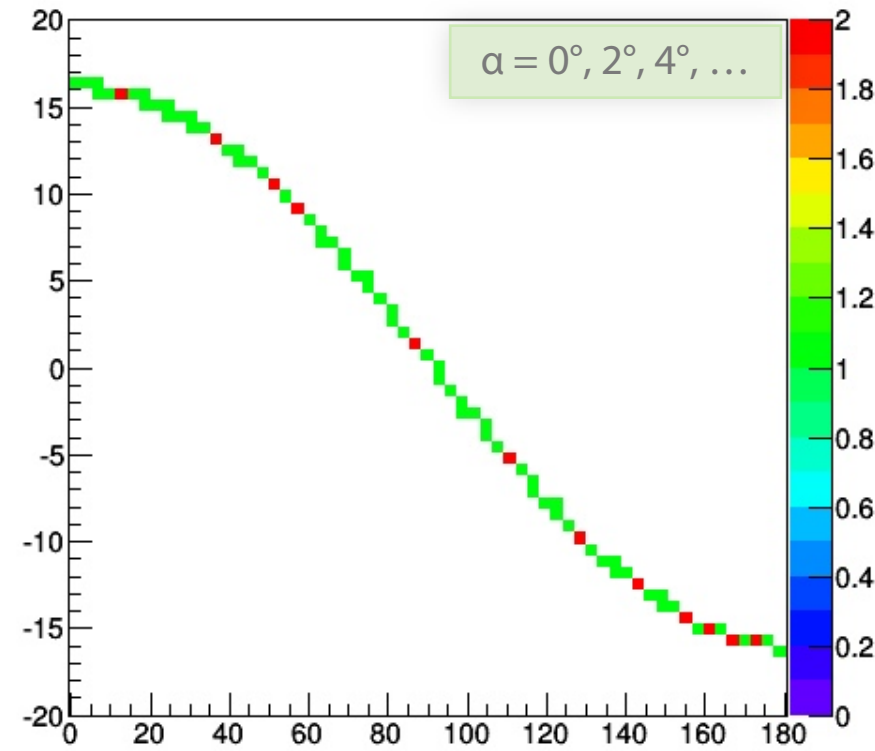
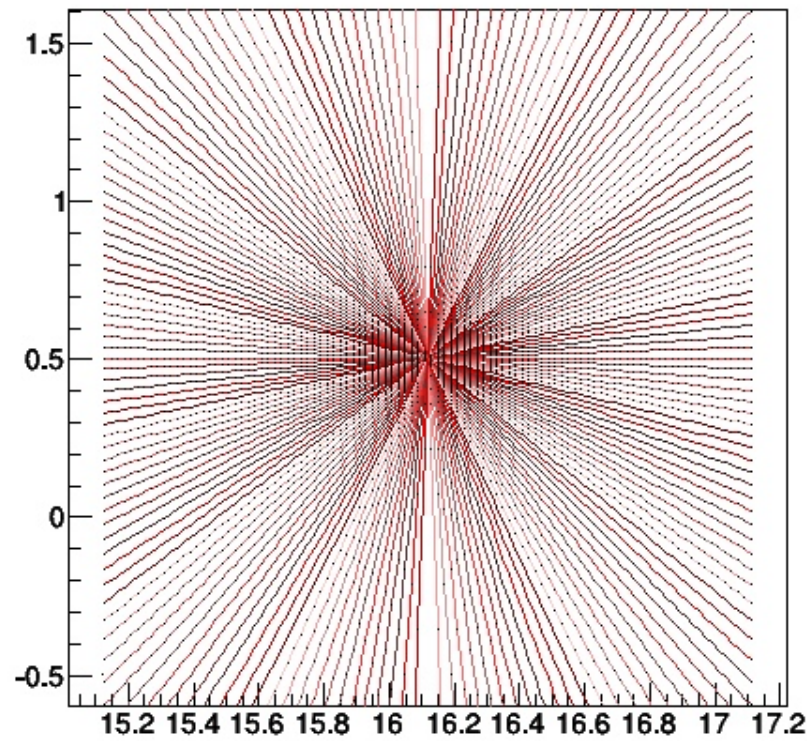
Hough Transform — Granularity

- Choice of α granularity determines resolution



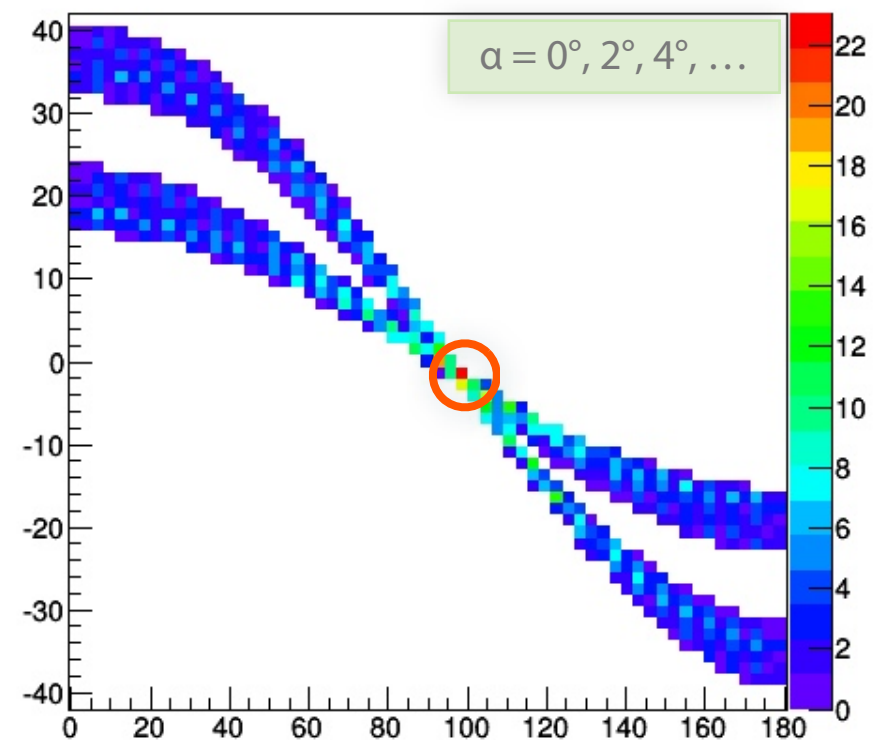
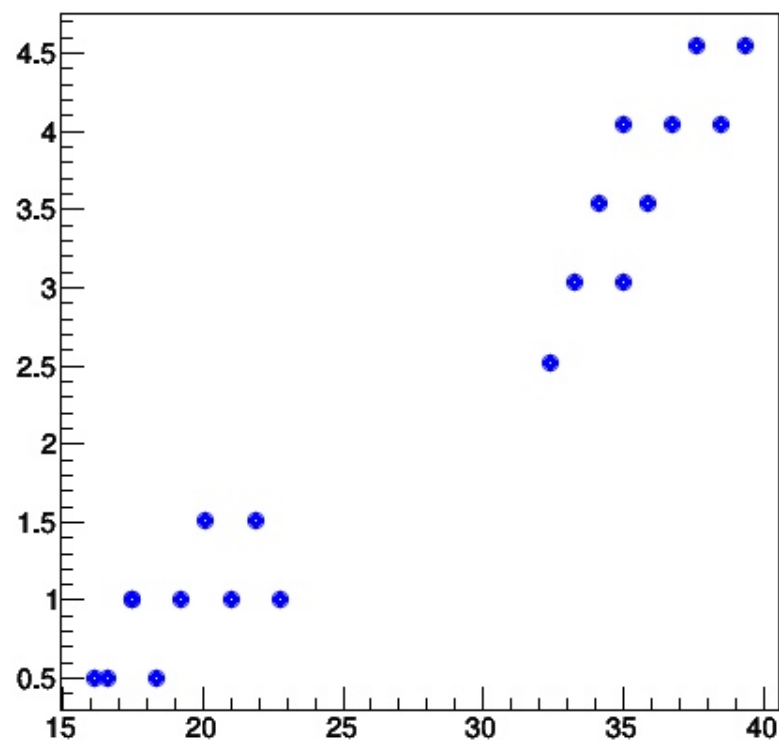
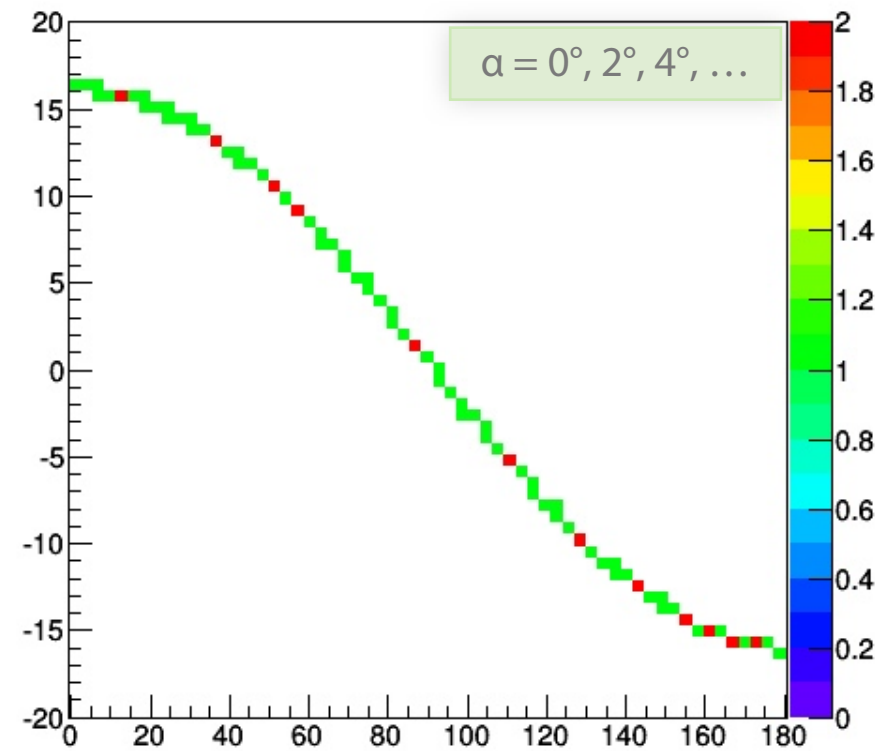
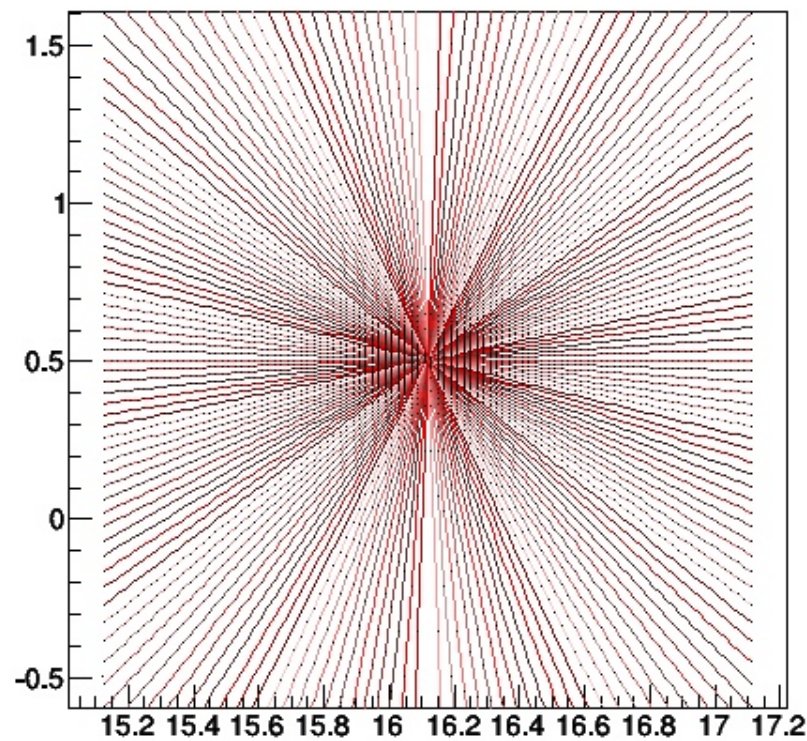
Hough Transform — Granularity

- Choice of α granularity determines resolution



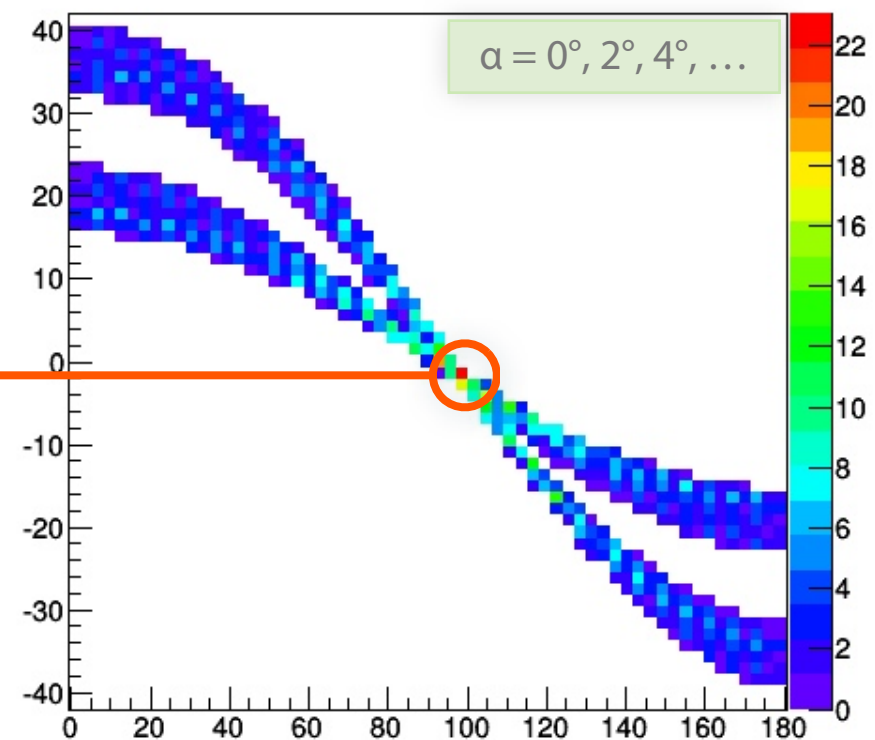
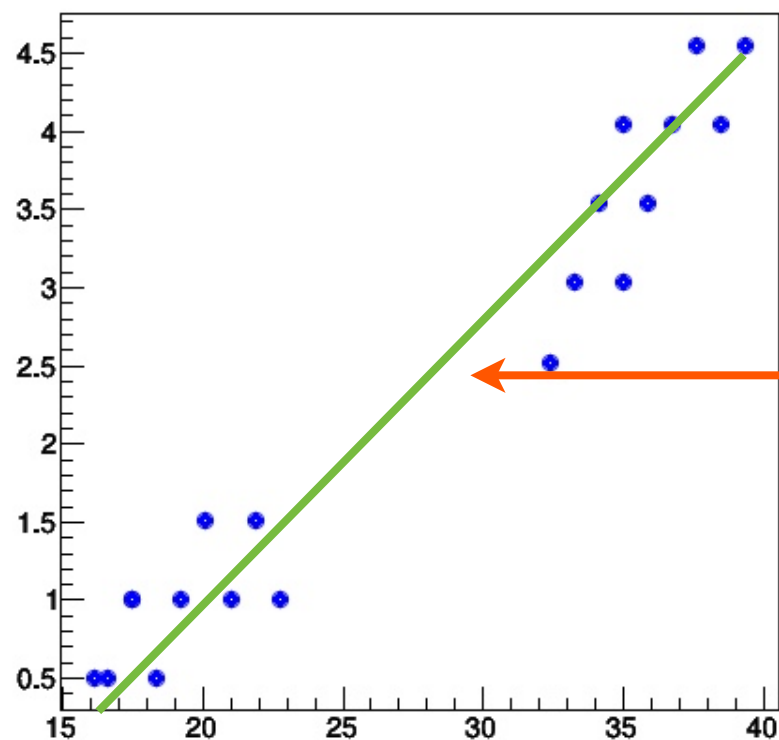
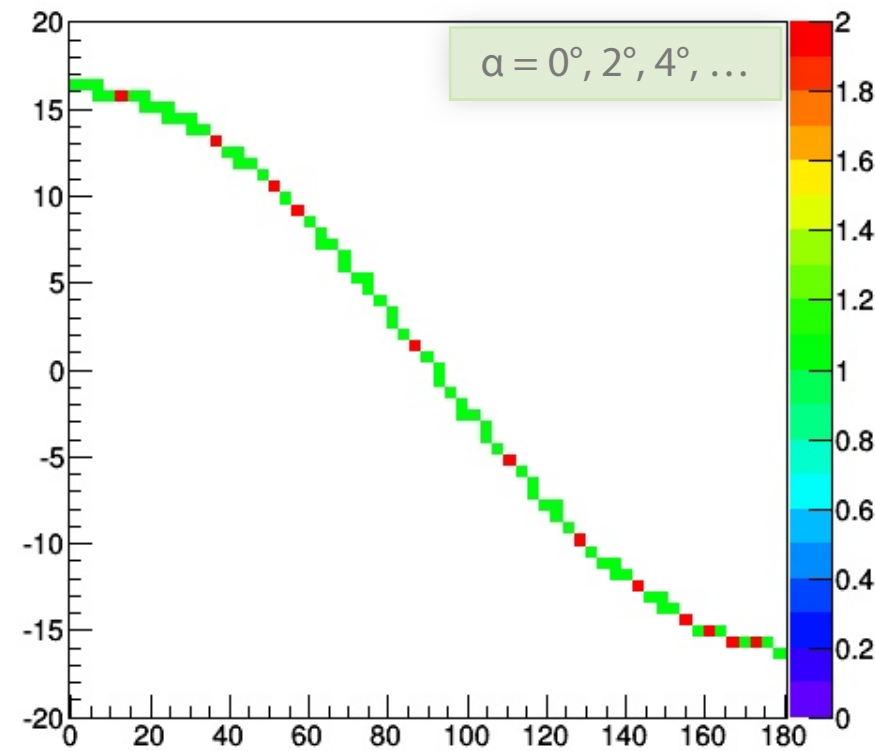
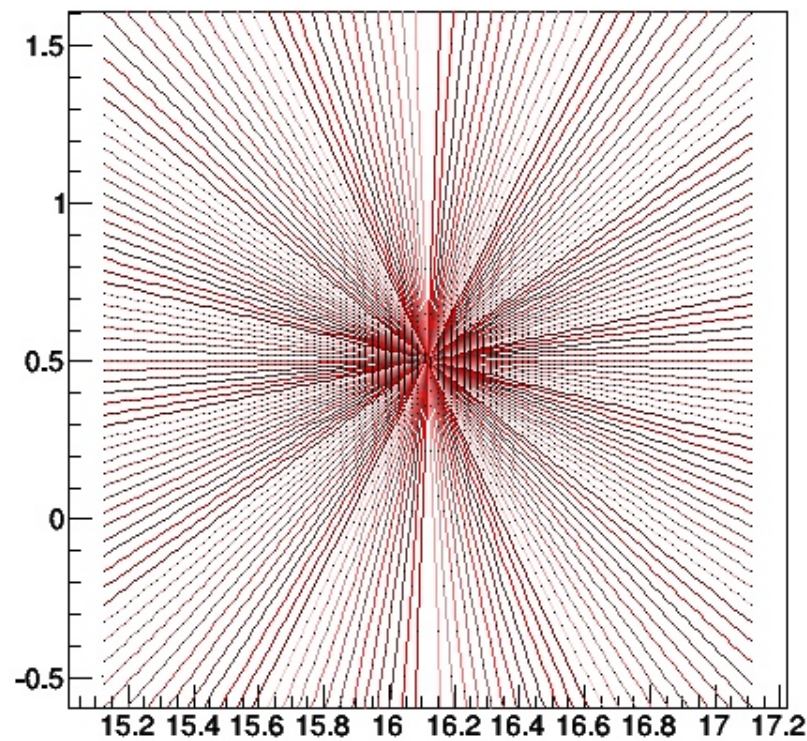
Hough Transform — Granularity

- Choice of α granularity determines resolution

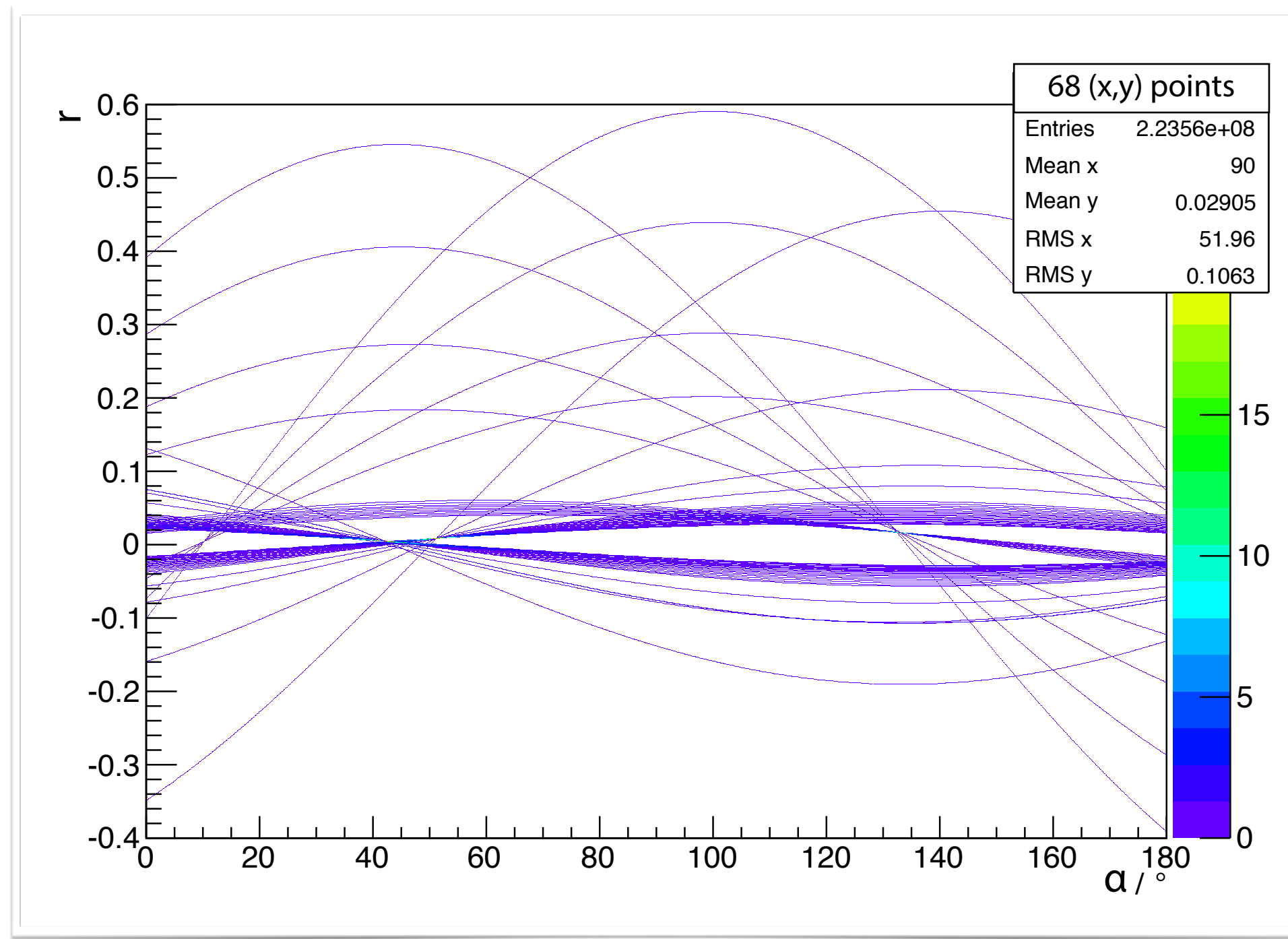


Hough Transform — Granularity

- Choice of α granularity determines resolution



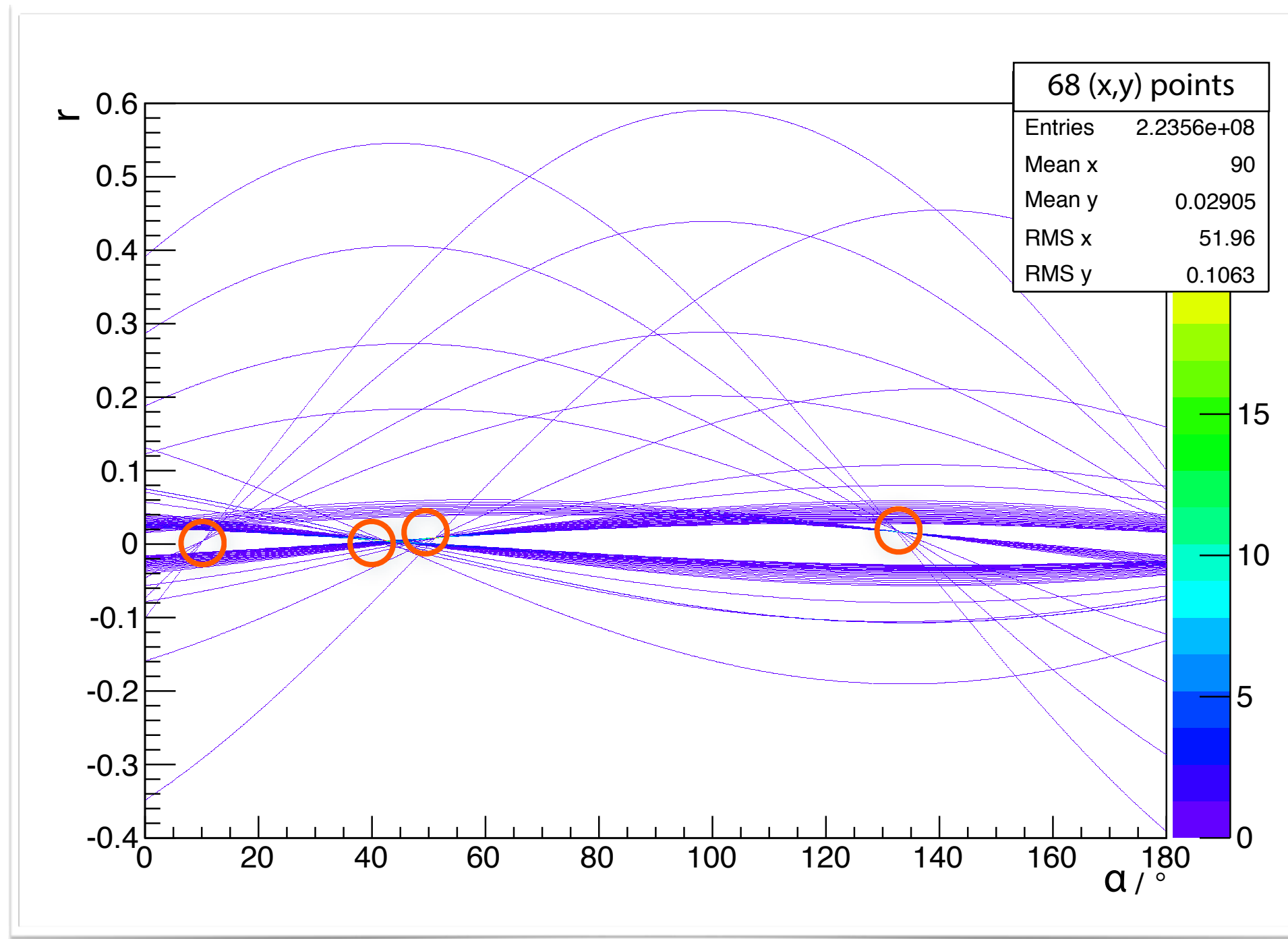
Algorithm: Hough Transform



PANDA STT+MVD

1800 x 1800 Grid

Algorithm: Hough Transform



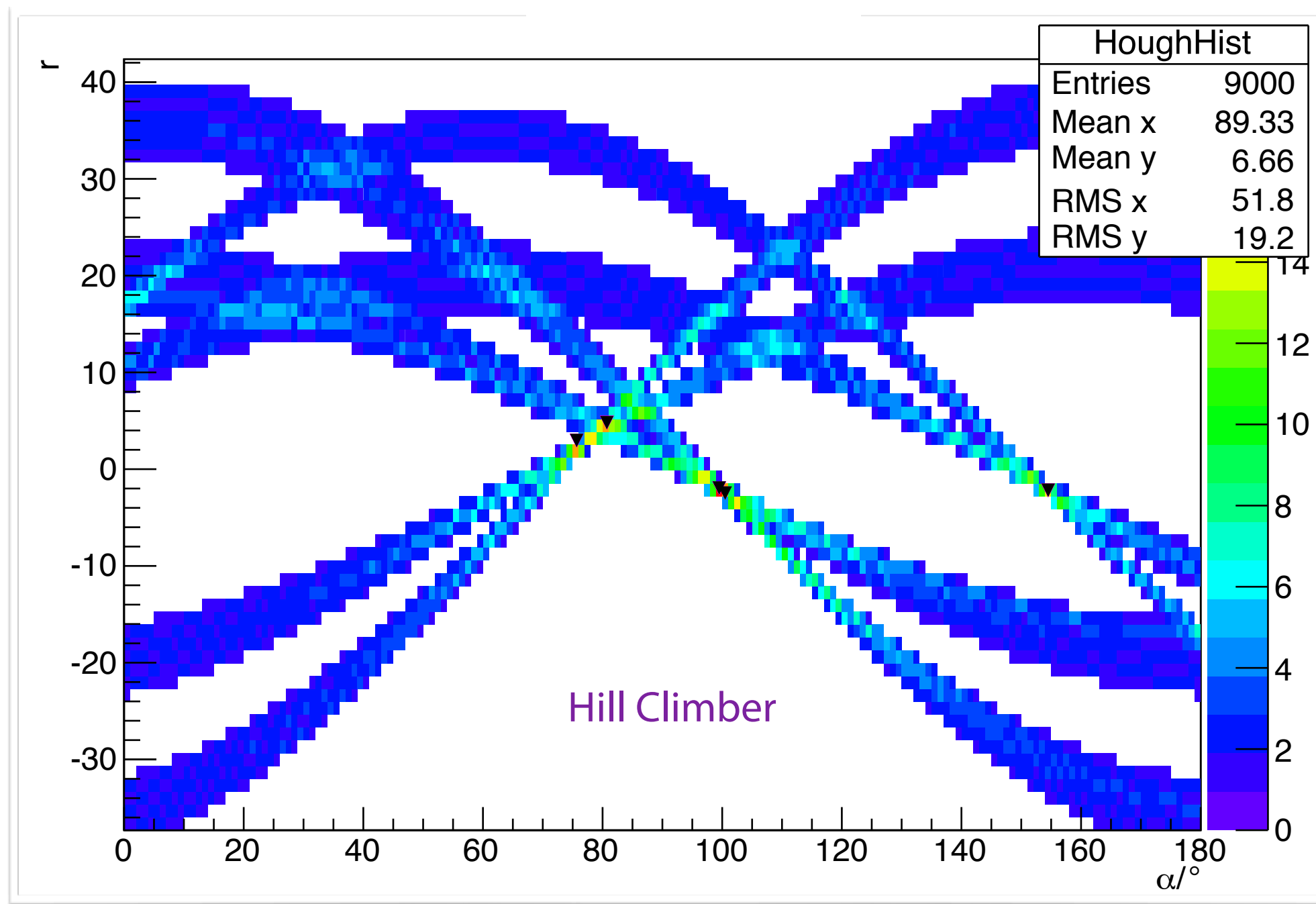
PANDA STT+MVD
1800 x 1800 Grid

Hough Transform — Remarks

Two Implementations

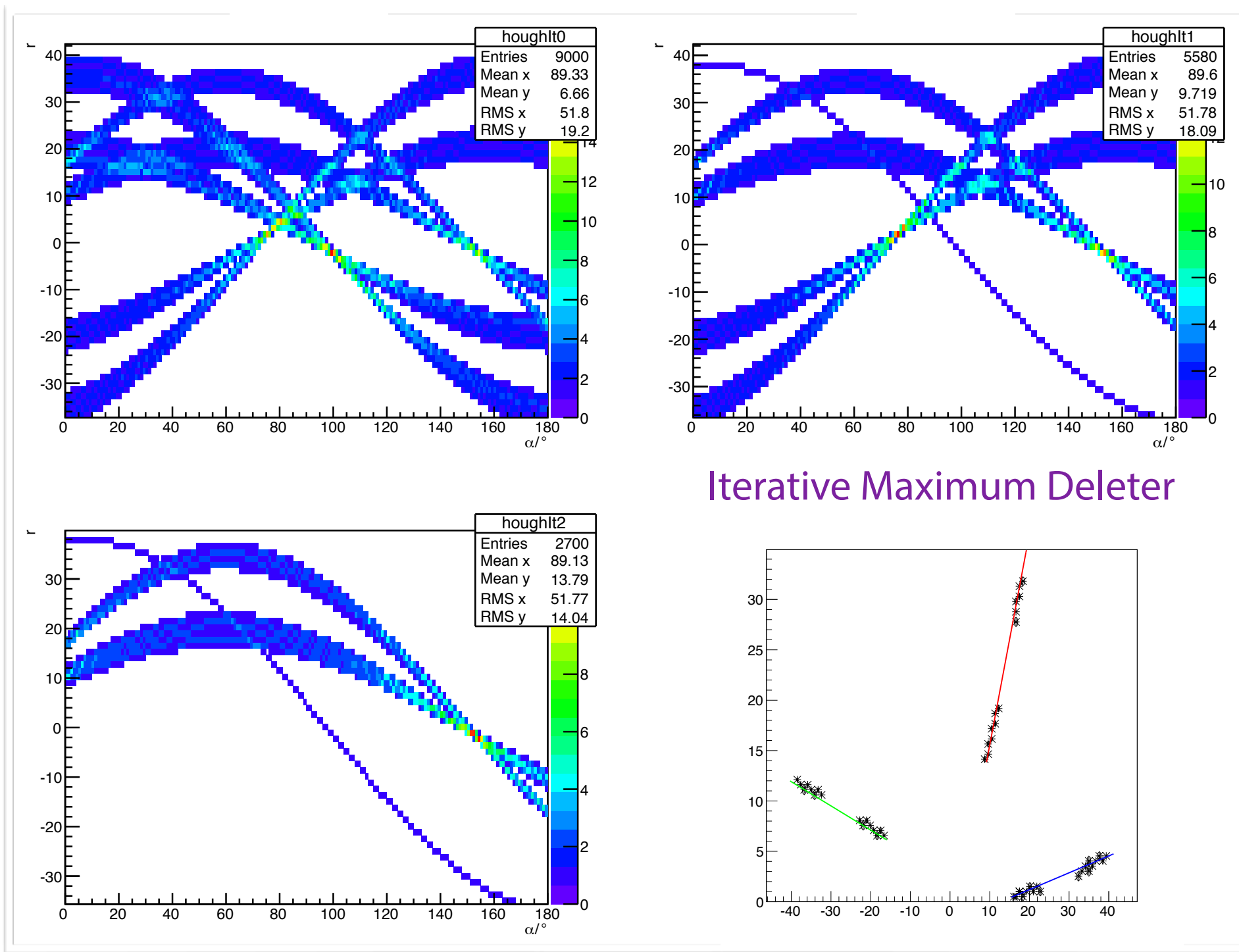
- Thrust (*CUDA's STL*)
 - **Performance:** 3 ms/evt
 - Reduce to set of **standard routines**
 - Fast (uses Thrust's optimized algorithms)
 - **Inflexible** (hard to customize)
 - Not yet at performance maximum
- Plain CUDA
 - **Performance:** 0.5 ms/evt
 - Built completely for this task
 - Fitting for PANDA; customizable
 - A bit more complicated at parts
- **Peakfinding** challenging

Hough Transform — Remarks



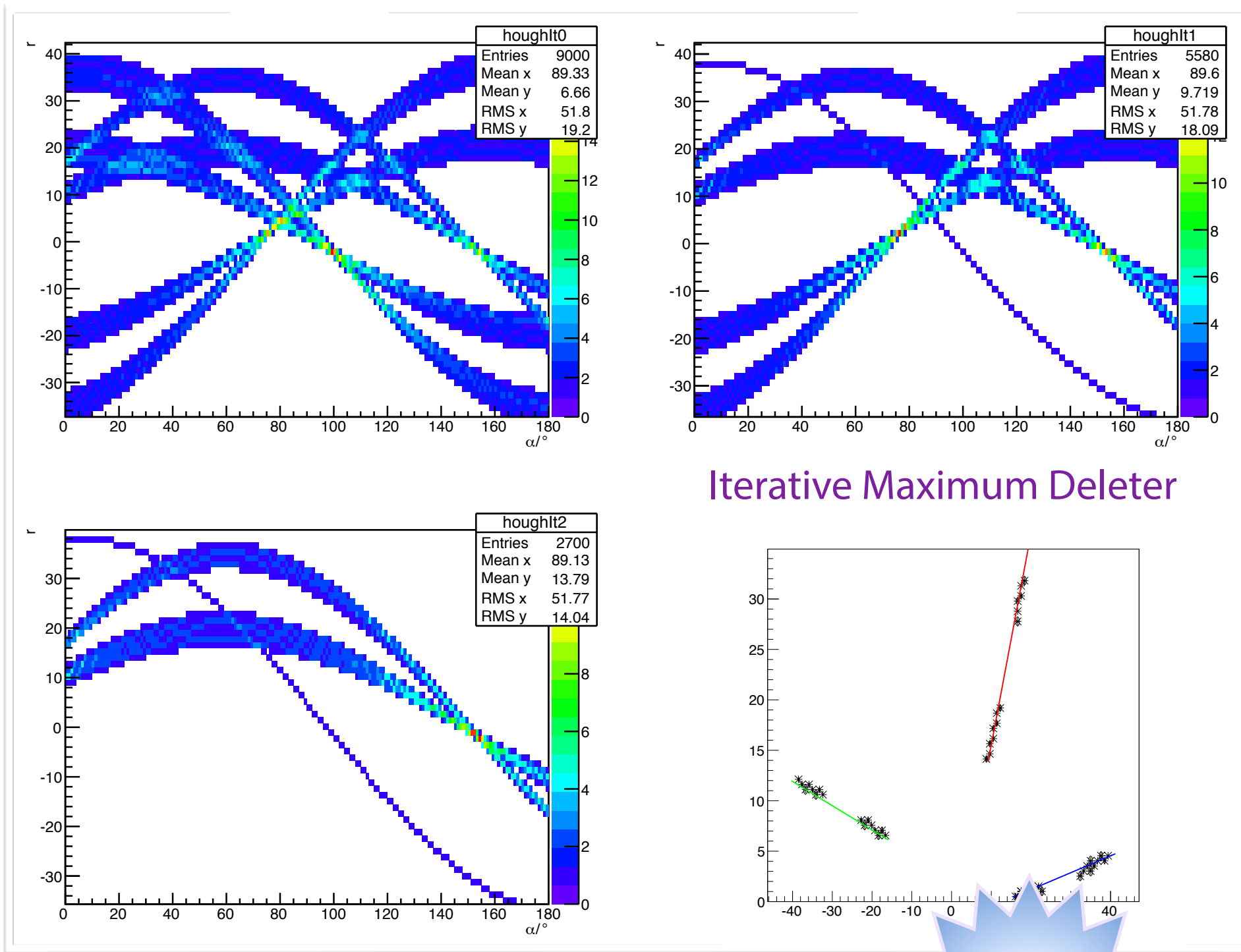
Peakfinding challenging

Hough Transform — Remarks



Peakfinding challenging

Hough Transform — Remarks



Iterative Maximum Deleter

Peakfinding challenging

current
research

ALGORITHMS #2

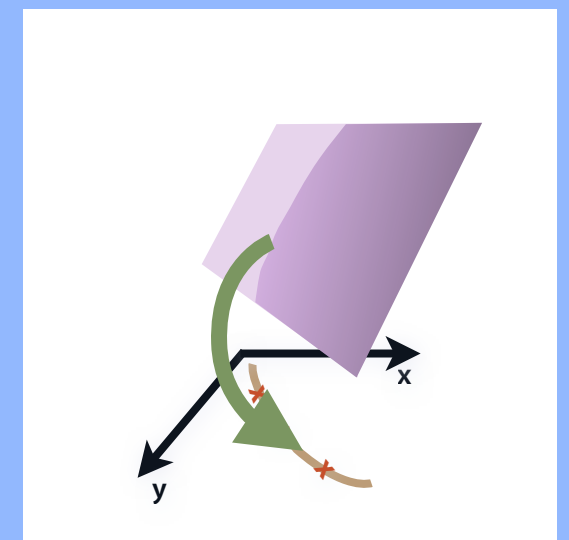
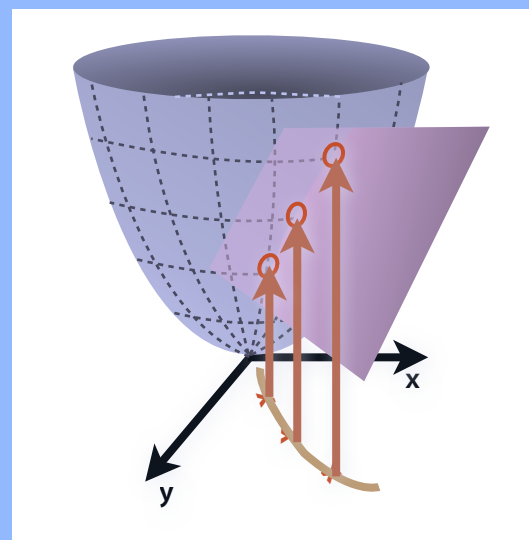
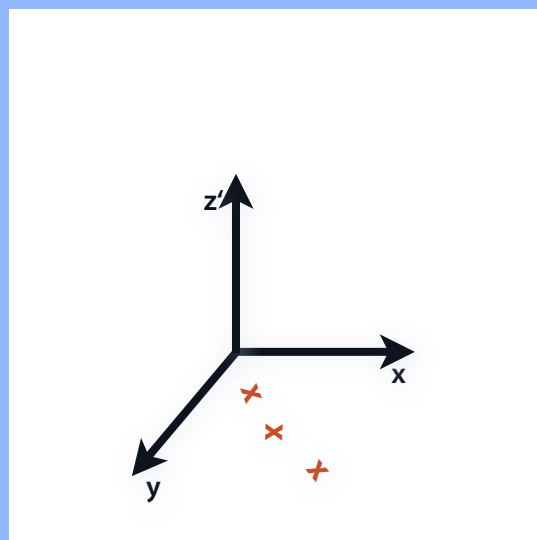
Hough Transform

Riemann Track Finder

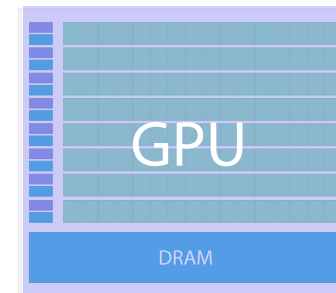
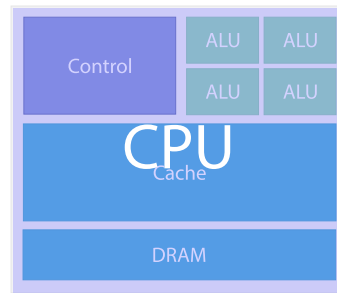
Triplet Finder

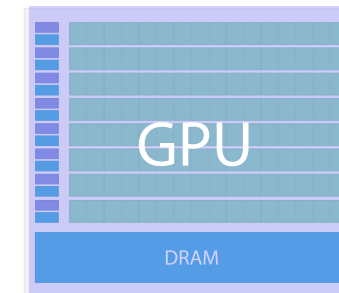
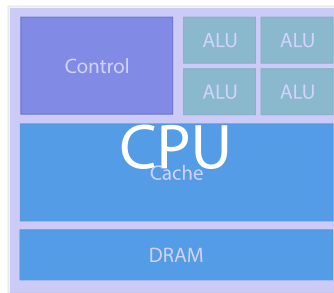
Riemann Track Finder — Method

- **Idea:** Don't fit lines (*in 2D*), fit planes (*in 3D*)!
- **Create seeds** 1
 - All possible three hit combinations
- **Grow seeds to tracks** 2
 - *Continuously test next hit if it fits*
 - Use mapping to Riemann paraboloid (+ *s-z fit, det. layer*)



Riemann Track Finder — GPU Adaptations





1

3 loops to generate seeds serially

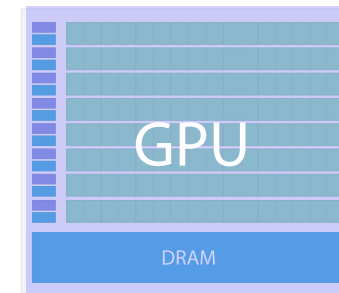
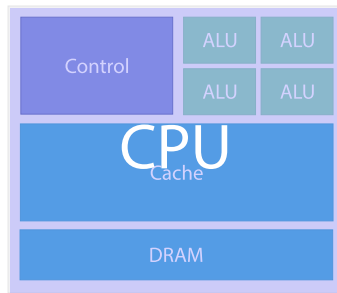
```
for (int i = 0; i < hitsInLayerOne.size(); i++) {  
    for (int j = 0; j < hitsInLayerTwo.size(); j++) {  
        for (int k = 0; k < hitsInLayerThree.size(); k++) {  
            /* Triplet Generation */  
        }  
    }  
}
```

Needed: Mapping of inherent GPU indexing variable to triplet index

```
int ijk = threadIdx.x + blockIdx.x * blockDim.x;
```

$$n_{\text{Layer}_x} = \frac{1}{2} (\sqrt{8x+1} - 1)$$

$$\text{pos}(n_{\text{Layer}_x}) = \frac{\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}}{3^{2/3}} + \frac{1}{\sqrt[3]{3}\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}} - 1$$



1

3 loops to generate seeds serially

```
for (int i = 0; i < hitsInLayerOne.size(); i++) {  
    for (int j = 0; j < hitsInLayerTwo.size(); j++) {  
        for (int k = 0; k < hitsInLayerThree.size(); k++) {  
            /* Triplet Generation */  
        }  
    }  
}
```

2

Only *easy* computations;
e.g. 3x3 matrices

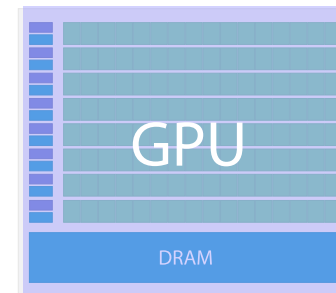
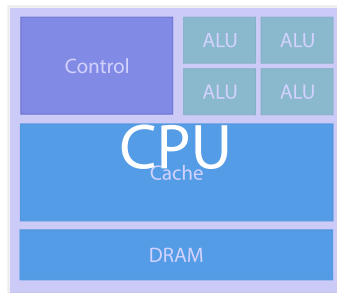
Needed: Mapping of
inherent GPU indexing
variable to triplet index

```
int ijk = threadIdx.x + blockIdx.x * blockDim.x;
```

$$n_{\text{Layer}_x} = \frac{1}{2} (\sqrt{8x+1} - 1)$$

$$\text{pos}(n_{\text{Layer}_x}) = \frac{\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}}{3^{2/3}} + \frac{1}{\sqrt[3]{3}\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}} - 1$$

Port of CPU code;
parallelism on seed base



1

3 loops to generate seeds serially

```
for (int i = 0; i < hitsInLayerOne.size(); i++) {  
    for (int j = 0; j < hitsInLayerTwo.size(); j++) {  
        for (int k = 0; k < hitsInLayerThree.size(); k++) {  
            /* Triplet Generation */  
        }  
    }  
}
```

2

Only *easy* computations;
e.g. 3x3 matrices

Needed: Mapping of
inherent GPU indexing
variable to triplet index

```
int ijk = threadIdx.x + blockIdx.x * blockDim.x;
```

$$n_{\text{Layer}_x} = \frac{1}{2} (\sqrt{8x+1} - 1)$$

$$\text{pos}(n_{\text{Layer}_x}) = \frac{\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}}{3^{2/3}} + \frac{1}{\sqrt[3]{3}\sqrt[3]{\sqrt{3}\sqrt{243x^2-1}+27x}} - 1$$

Port of CPU code;
parallelism on seed base

→ 100 × faster than CPU version: ~0.6 ms/event

Still needs implementing into PandaRoot

ALGORITHMS #3

Hough Transform

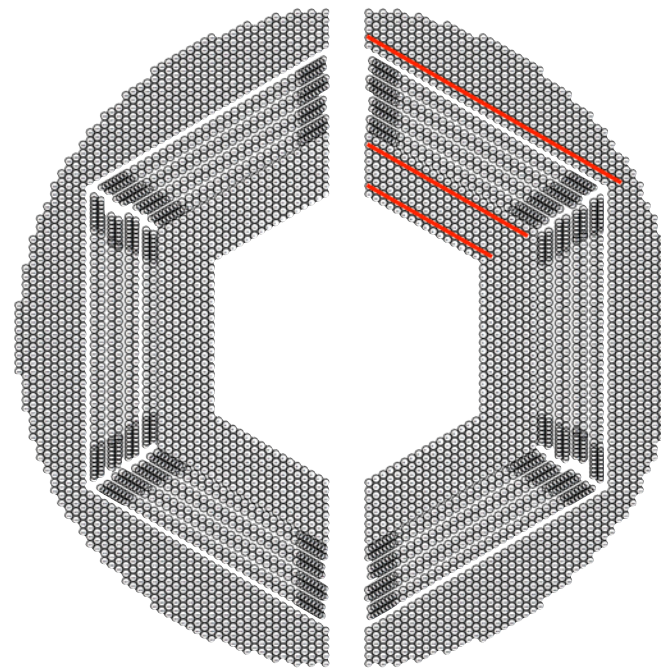
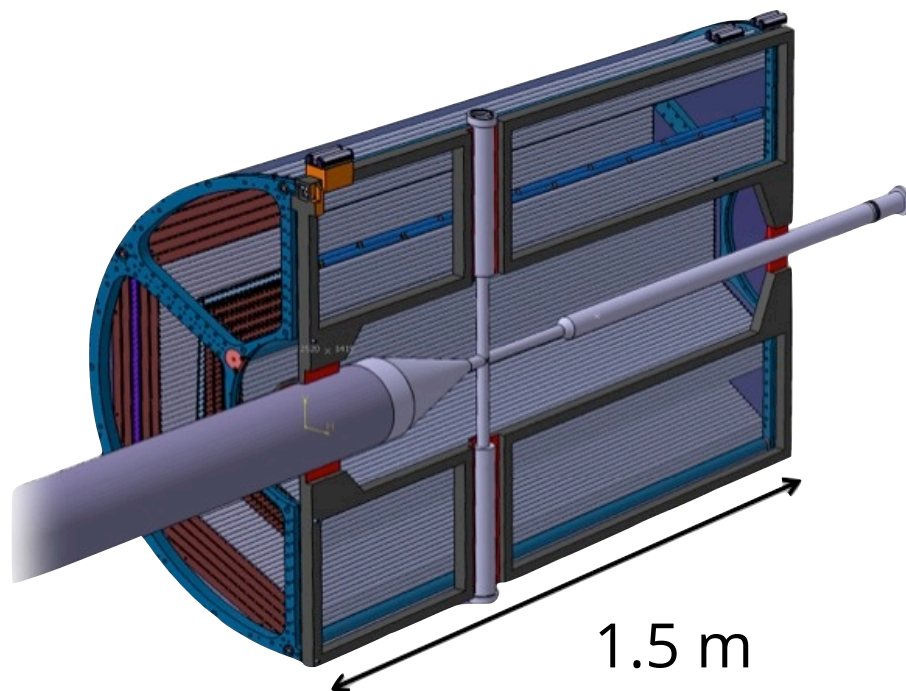
Riemann Track Finder

Triplet Finder

Triplet Finder

- Algorithm specifically designed for the $\bar{\text{P}}$ ANDA Straw Tube Tracker (*STT*)

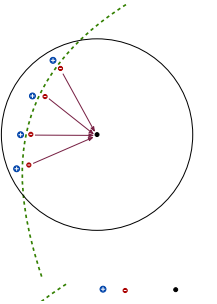
Original algorithm by
Marius Mertens *et al*



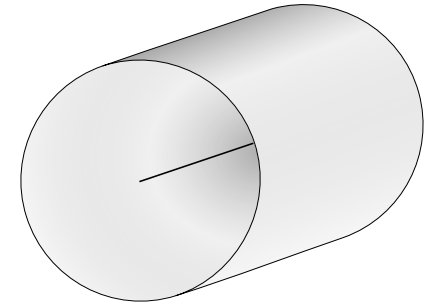
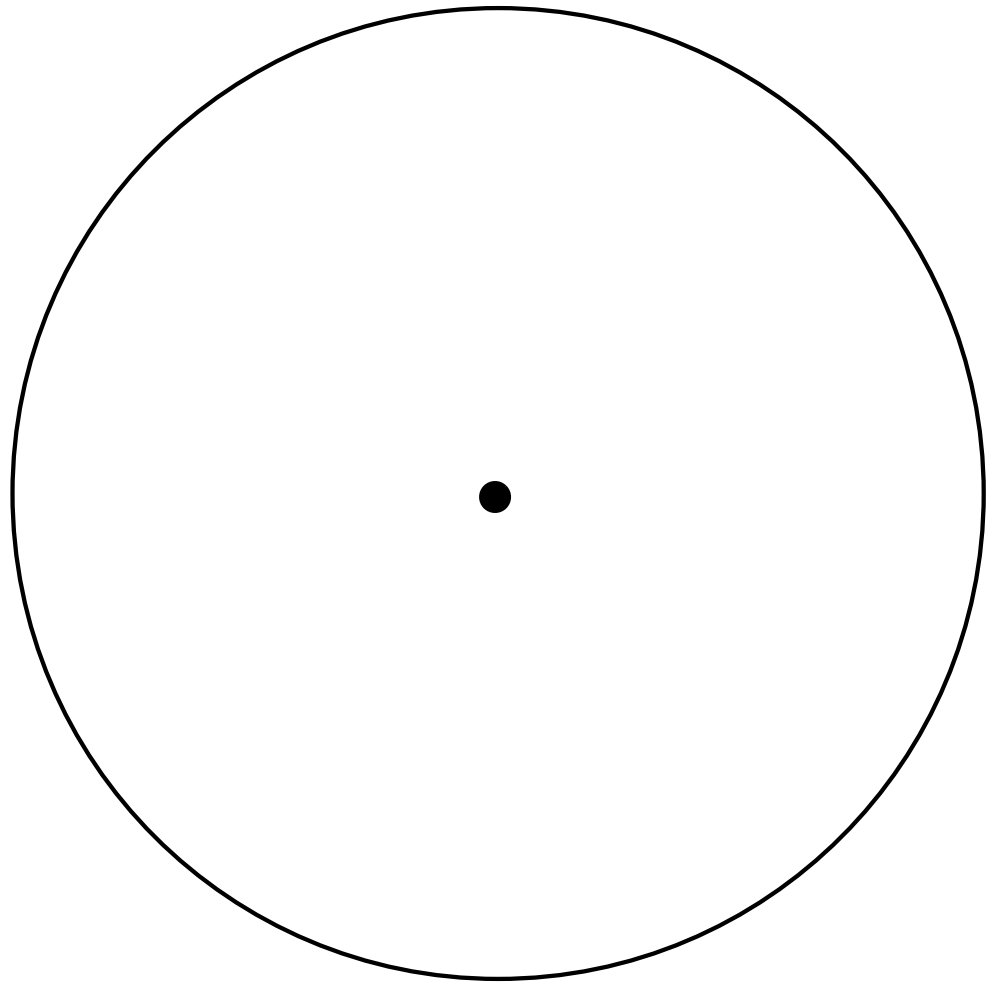
- Ported to GPU by **Andrew Adinetz**
 - CUDA, Dynamic Parallelism, Thrust
 - Quality of tracks comparable to CPU

Triplet Finder

- **Idea:** Use only subset of detector as seed
 - Don't use STT isochrones (*drift times*)
 - Calculate circle from 3 points (no fit)

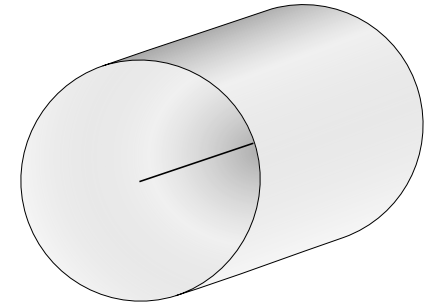
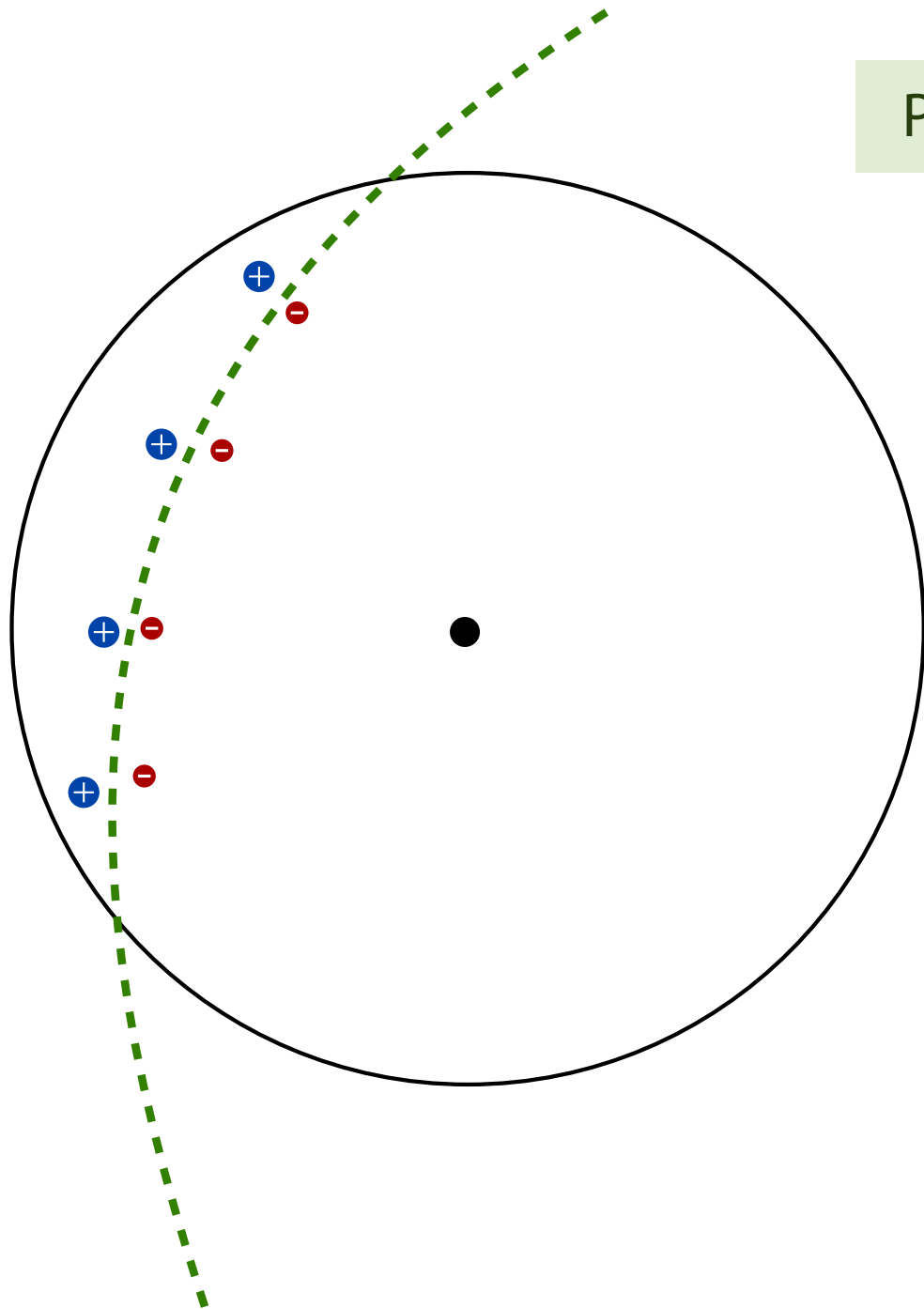


STT — Drift Tubes and t_0



STT — Drift Tubes and t_0

Particle ionizes gas atoms in drift tubes

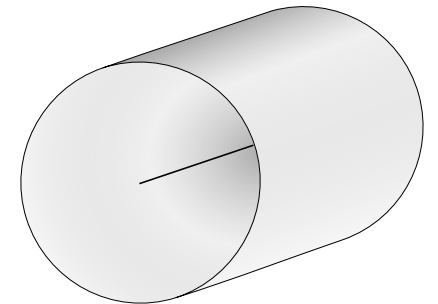
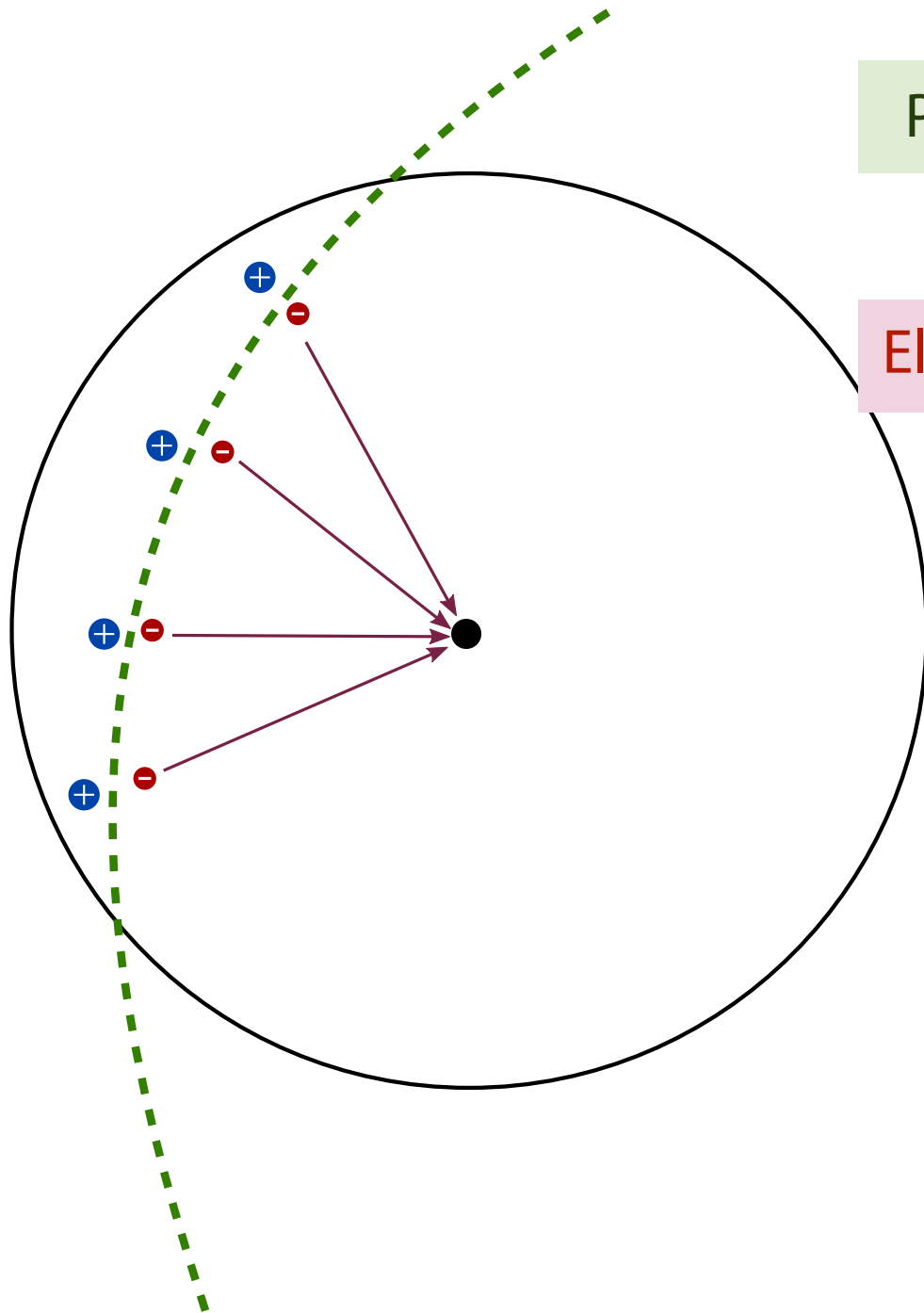


STT — Drift Tubes and t_0

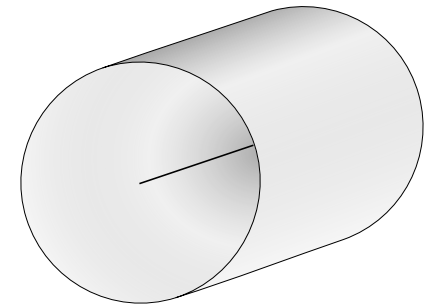
Particle ionizes gas atoms in drift tubes



Electrons drift to anode wire, ions to wall



STT — Drift Tubes and t_0



Particle ionizes gas atoms in drift tubes



Electrons drift to anode wire, **ions** to wall

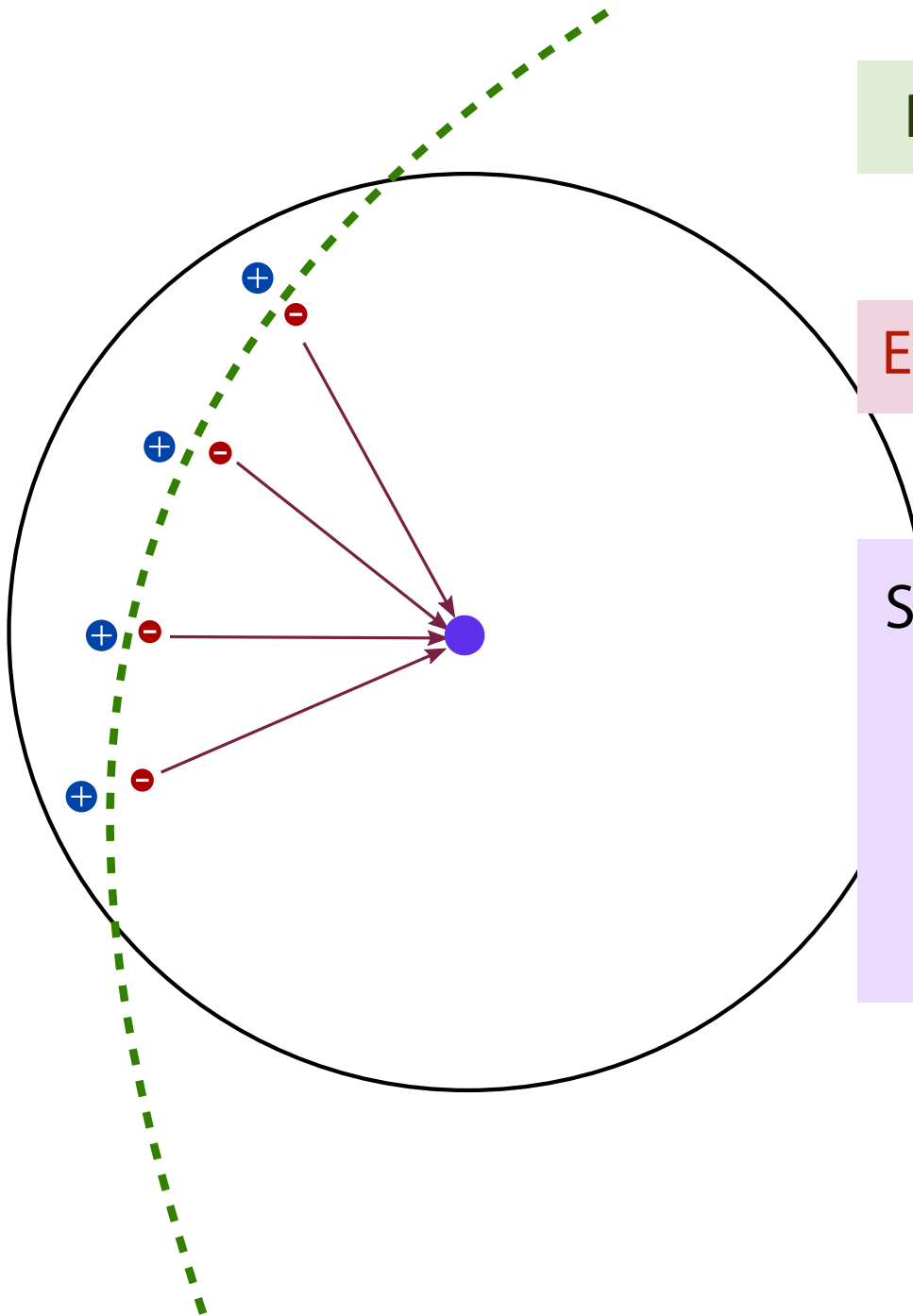


Signal only when **electrons** arrive at wire
No information about drift duration!

For that, start time (t_0) needed:

$$t_0 - t_{\text{arrival}} \approx t_{\text{drift}}$$

$$v_{\text{drift}} = \text{const} \rightarrow t_{\text{drift}} \cdot v_{\text{drift}} = r_{\text{isochrone}}$$



STT — Drift Tubes and t_0

Particle ionizes gas atoms in drift tubes



Electrons drift to anode wire, **ions** to wall

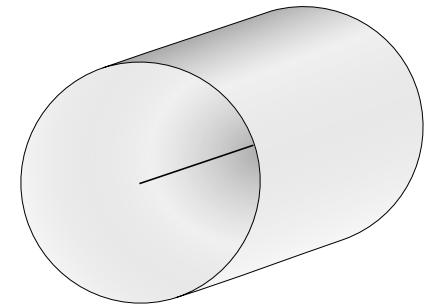
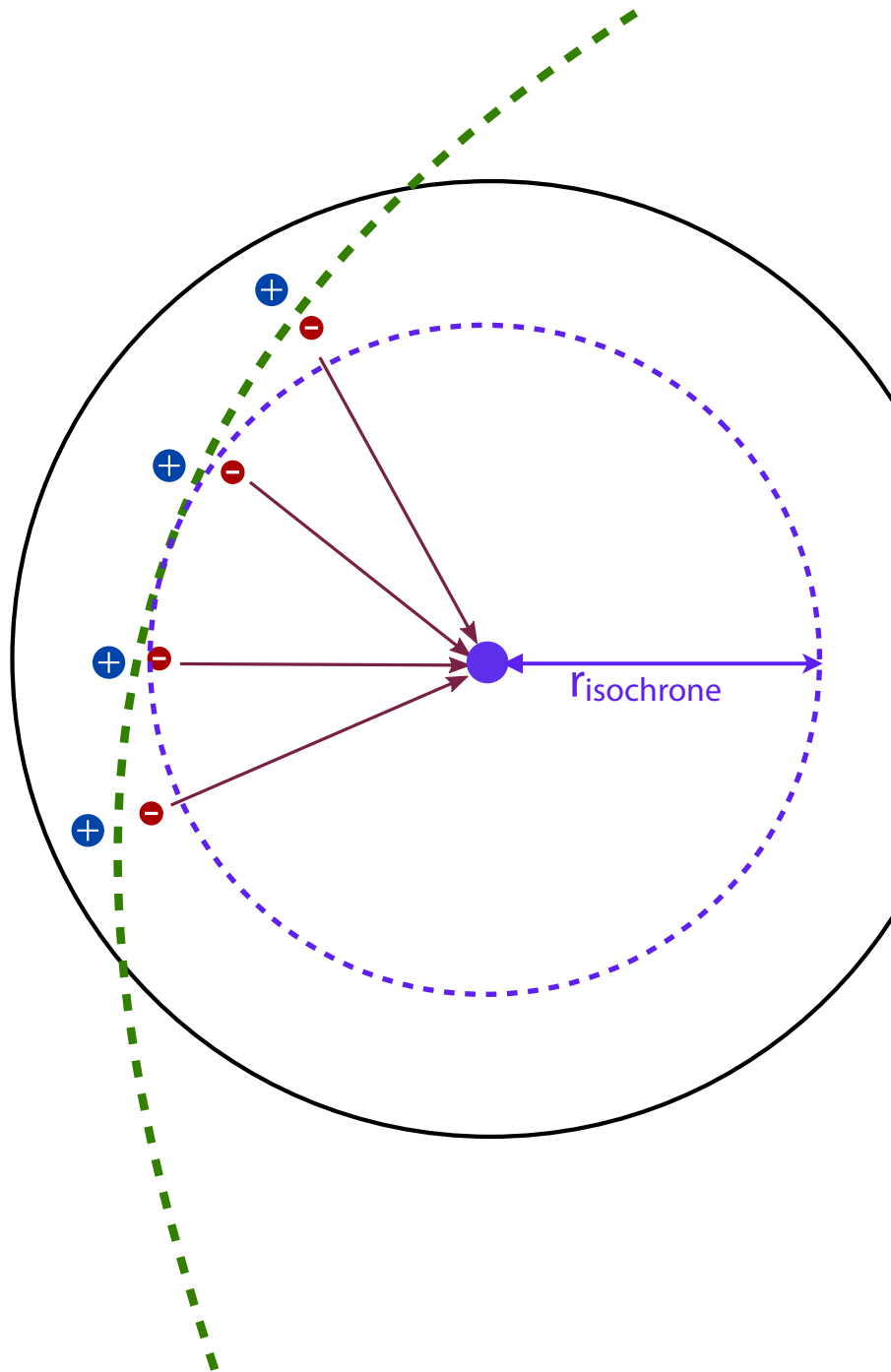


Signal only when **electrons** arrive at wire
No information about drift duration!

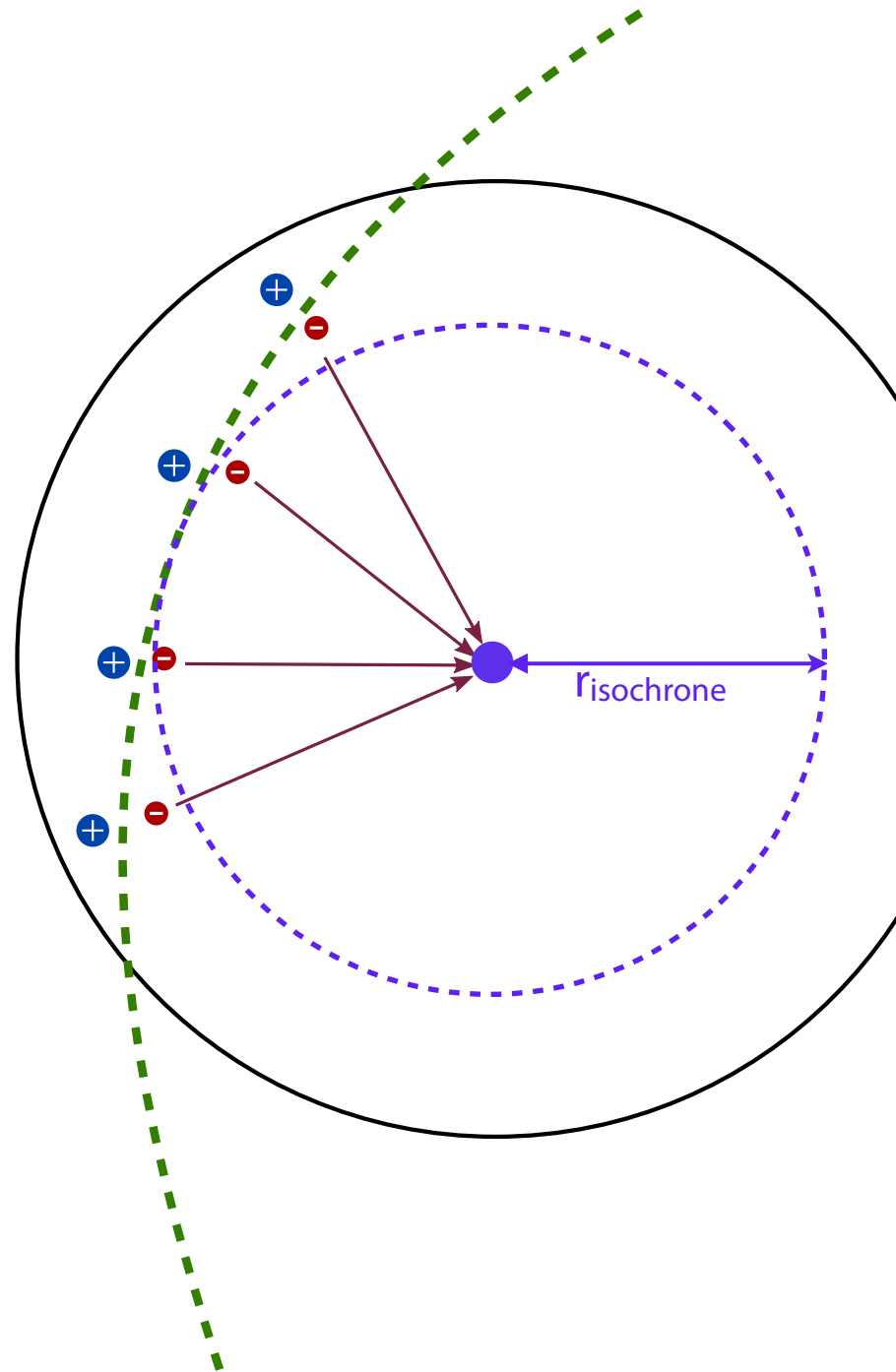
For that, start time (t_0) needed:

$$t_0 - t_{\text{arrival}} \approx t_{\text{drift}}$$

$$v_{\text{drift}} = \text{const} \rightarrow t_{\text{drift}} \cdot v_{\text{drift}} = r_{\text{isochrone}}$$



STT — Drift Tubes and t_0



Particle ionizes gas atoms in drift tubes



Electrons drift to anode wire, **ions** to wall



Signal only when **electrons** arrive at wire
No information about drift duration!

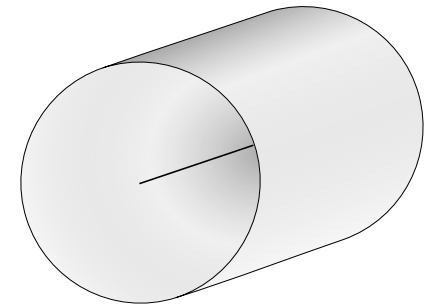
For that, start time (t_0) needed:

$$t_0 - t_{\text{arrival}} \approx t_{\text{drift}}$$

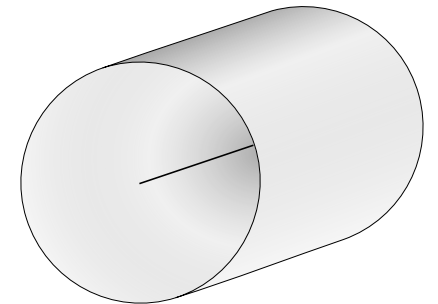
$$v_{\text{drift}} = \text{const} \rightarrow t_{\text{drift}} \cdot v_{\text{drift}} = r_{\text{isochrone}}$$

Resolution without t_0 : $\mathcal{O}(0.1 \text{ cm})$ (r_{tube})

Resolution with t_0 : $\mathcal{O}(0.015 \text{ cm})$



STT — Drift Tubes and t_0



Particle ionizes gas atoms in drift tubes



Electrons drift to anode wire, **ions** to wall



Signal only when **electrons** arrive at wire
No information about drift duration!

For that, start time (t_0) needed:

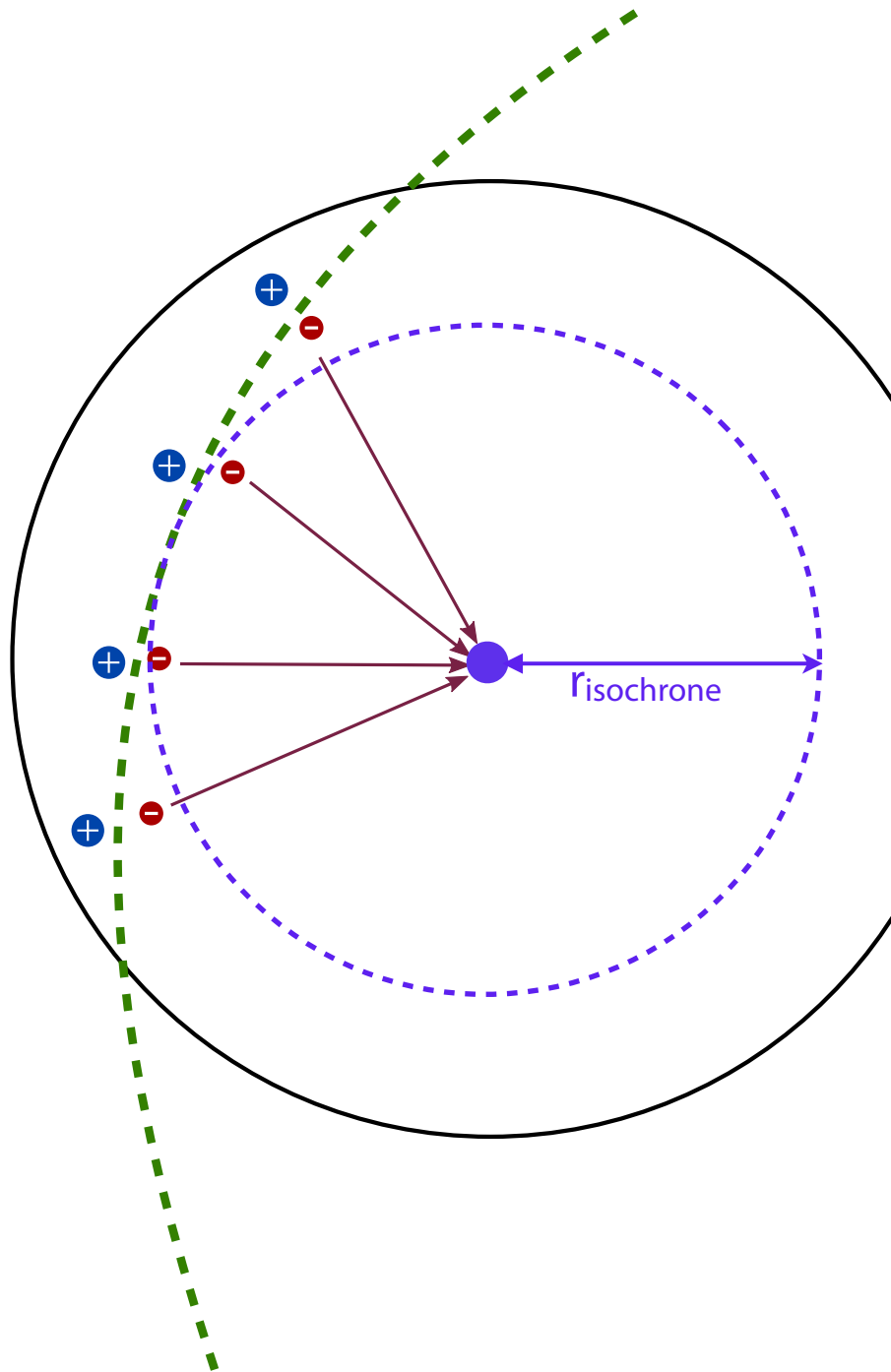
$$t_0 - t_{\text{arrival}} \approx t_{\text{drift}}$$

$$v_{\text{drift}} = \text{const} \rightarrow t_{\text{drift}} \cdot v_{\text{drift}} = r_{\text{isochrone}}$$

Resolution without t_0 : $\mathcal{O}(0.1 \text{ cm})$ (r_{tube})

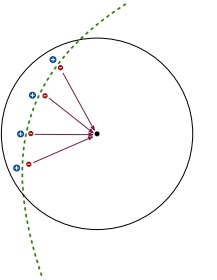
Resolution with t_0 : $\mathcal{O}(0.015 \text{ cm})$

Usual HEP experiment: t_0 by trigger
But PANDA has no trigger...



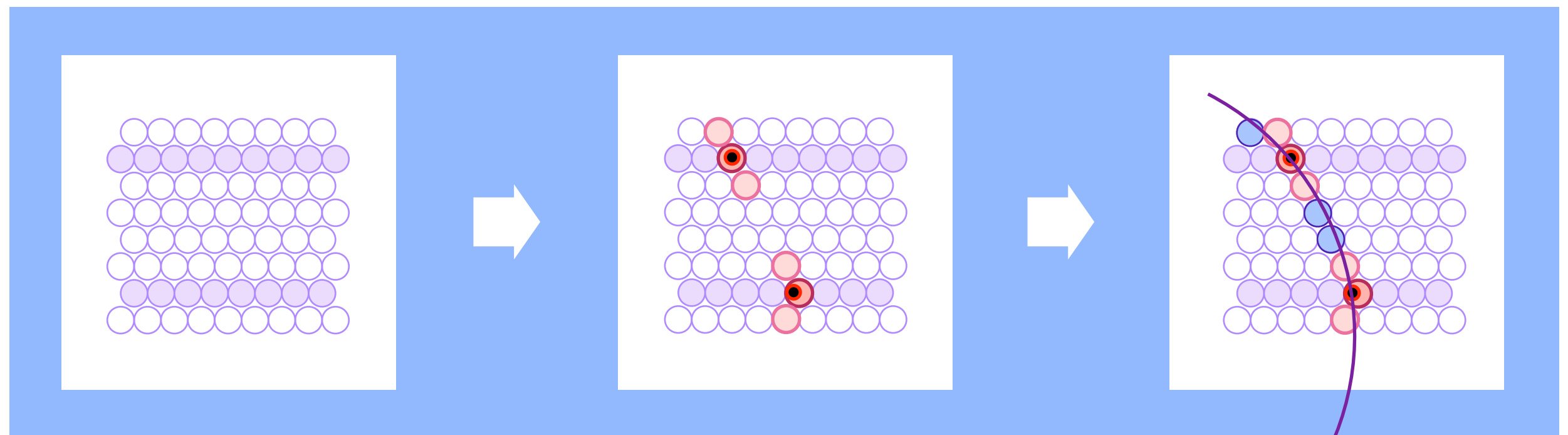
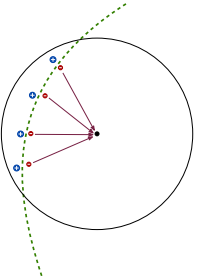
Triplet Finder

- **Idea:** Use only subset of detector as seed
 - Don't use STT isochrones (*drift times*)
 - Calculate circle from 3 points (no fit)

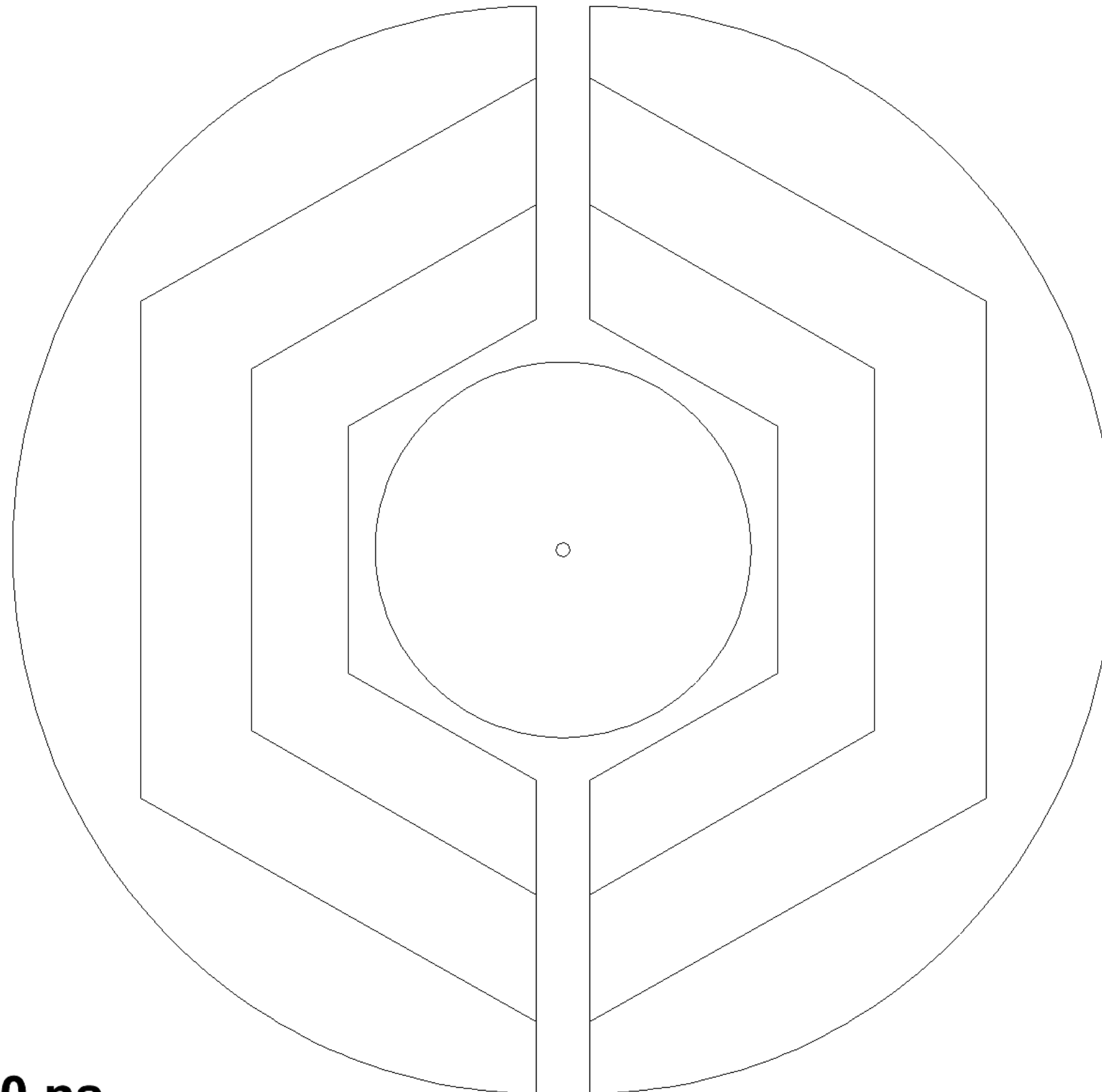


Triplet Finder

- **Idea:** Use only subset of detector as seed
 - Don't use STT isochrones (*drift times*)
 - Calculate circle from 3 points (no fit)
- Features
 - Fast & robust algorithm, no event time needed
 - Many tuning possibilities



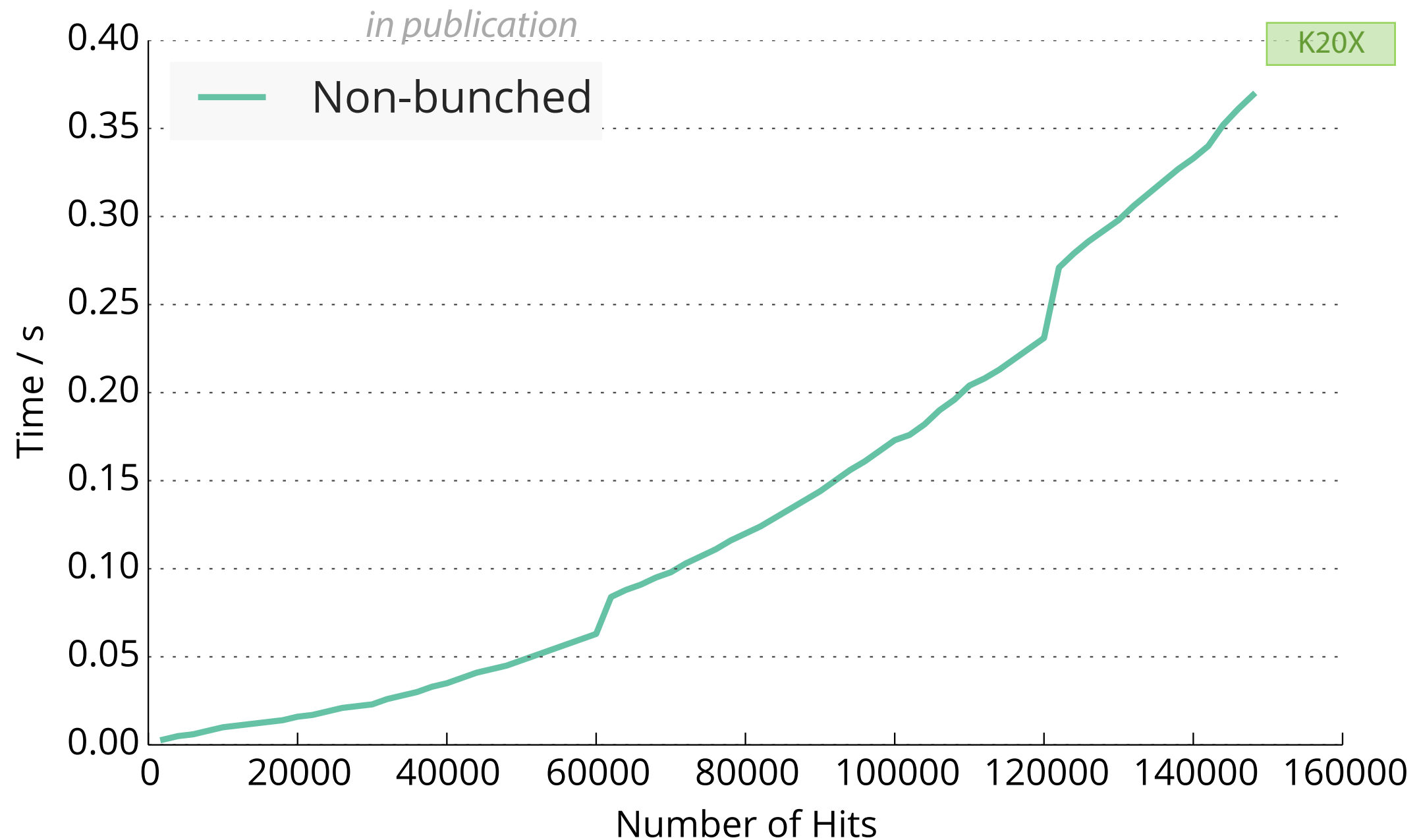
Triplet Finder — Animation



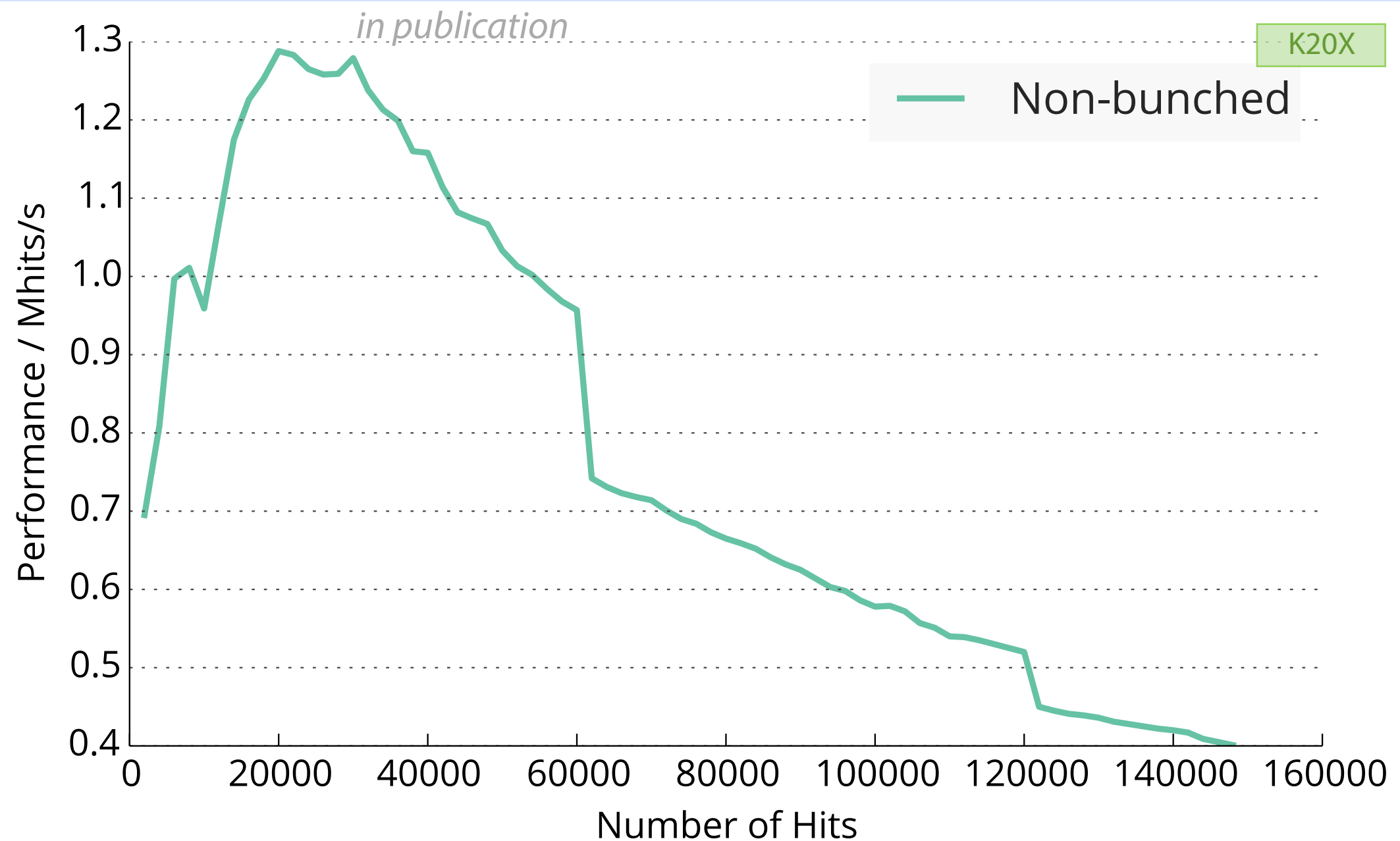
0 ns

- Isochrone *early*
- Isochrone *early* & *skewed*
- Isochrone *close*
- Isochrone *late*
- MVD hit
- Triplet
- Track *current*
- Track *timed out*

Triplet Finder — Times



Triplet Finder — Times



Triplet Finder — Optimizations

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

Hit



Triplet Finder — Optimizations

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

Hit

Event



Triplet Finder — Optimizations

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

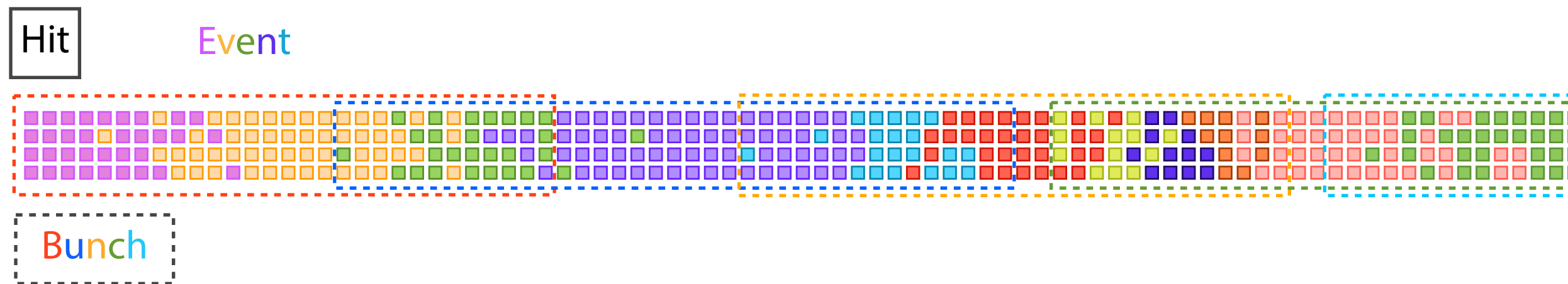
Hit

Event



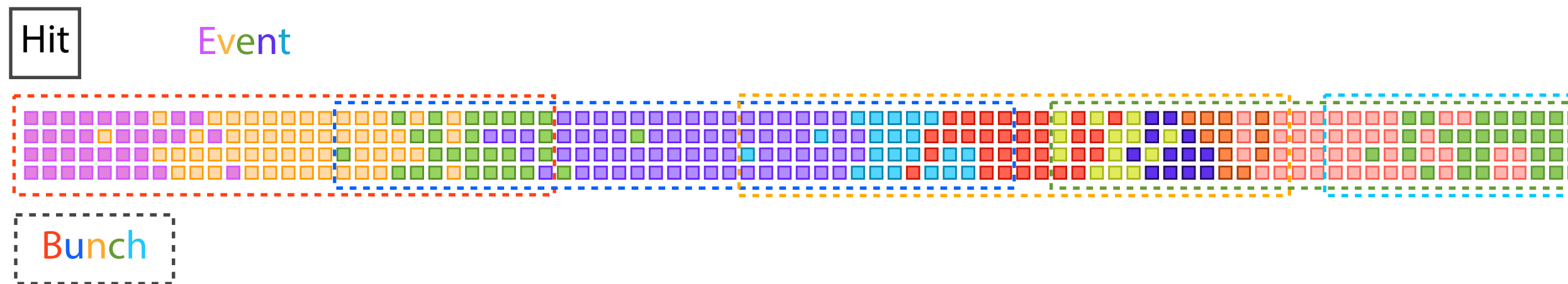
Triplet Finder — Optimizations

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

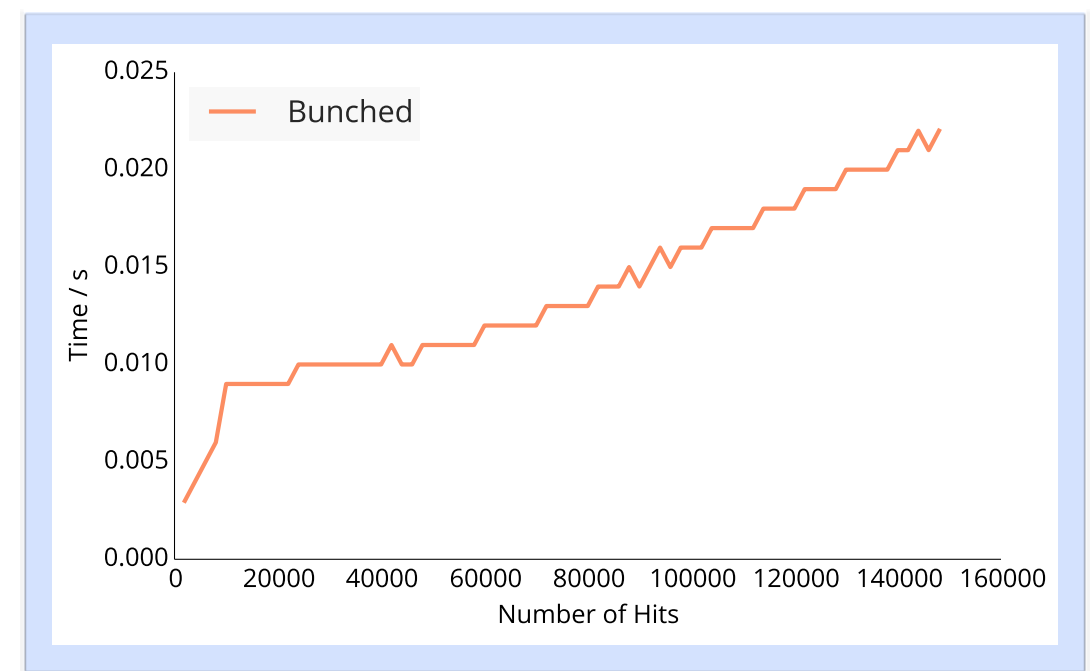
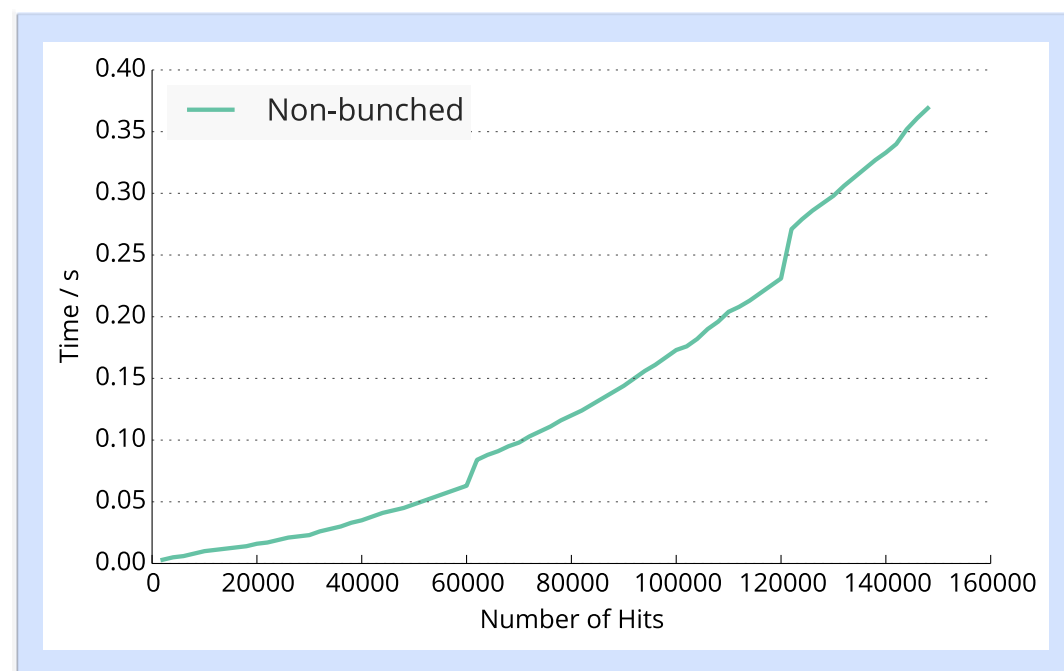


Triplet Finder — Optimizations

- Bunching Wrapper
 - Hits from one event have similar timestamps
 - Combine hits to sets (bunches) which occupy GPU best

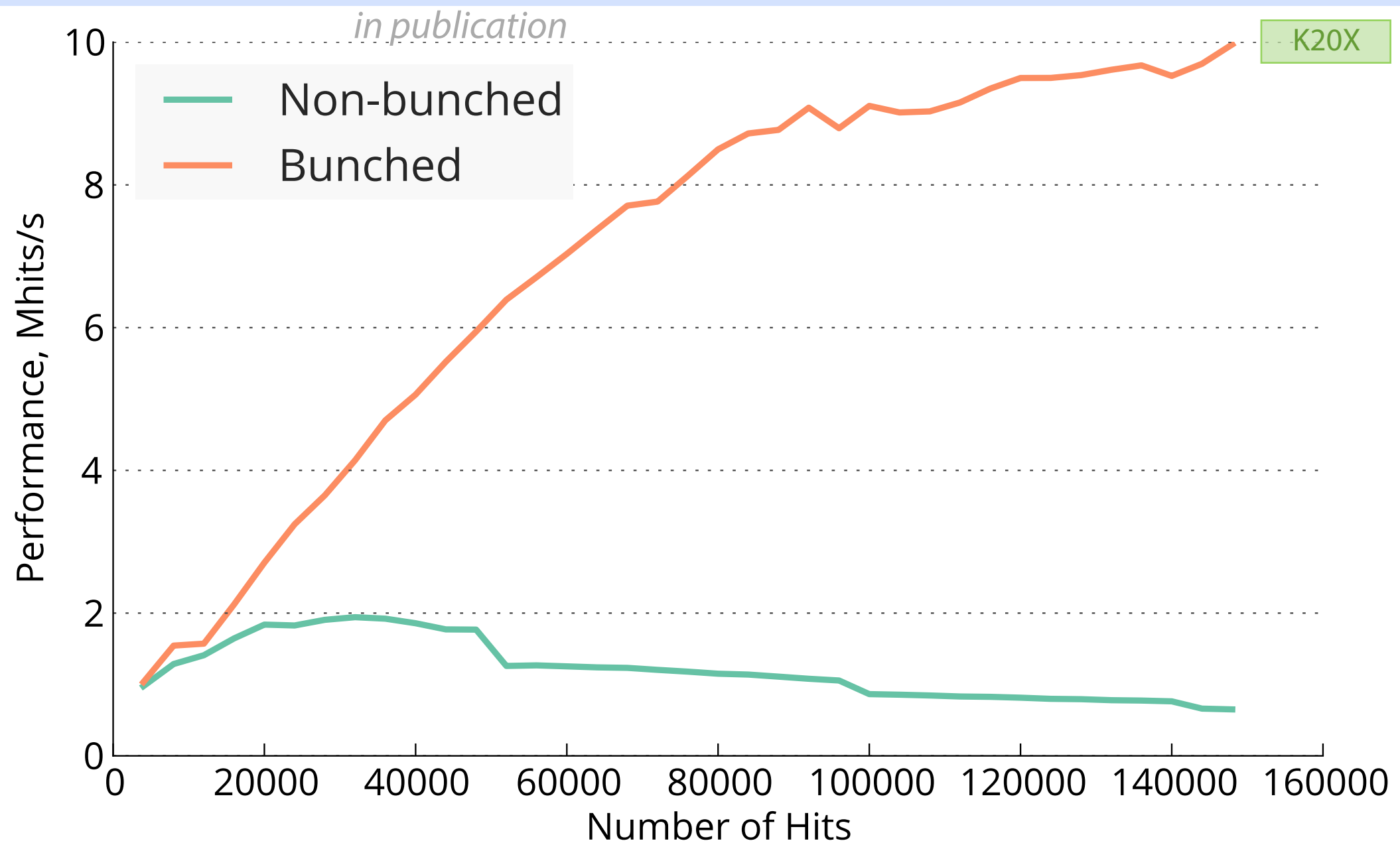


$$\mathcal{O}(N^2) \rightarrow \mathcal{O}(N)$$



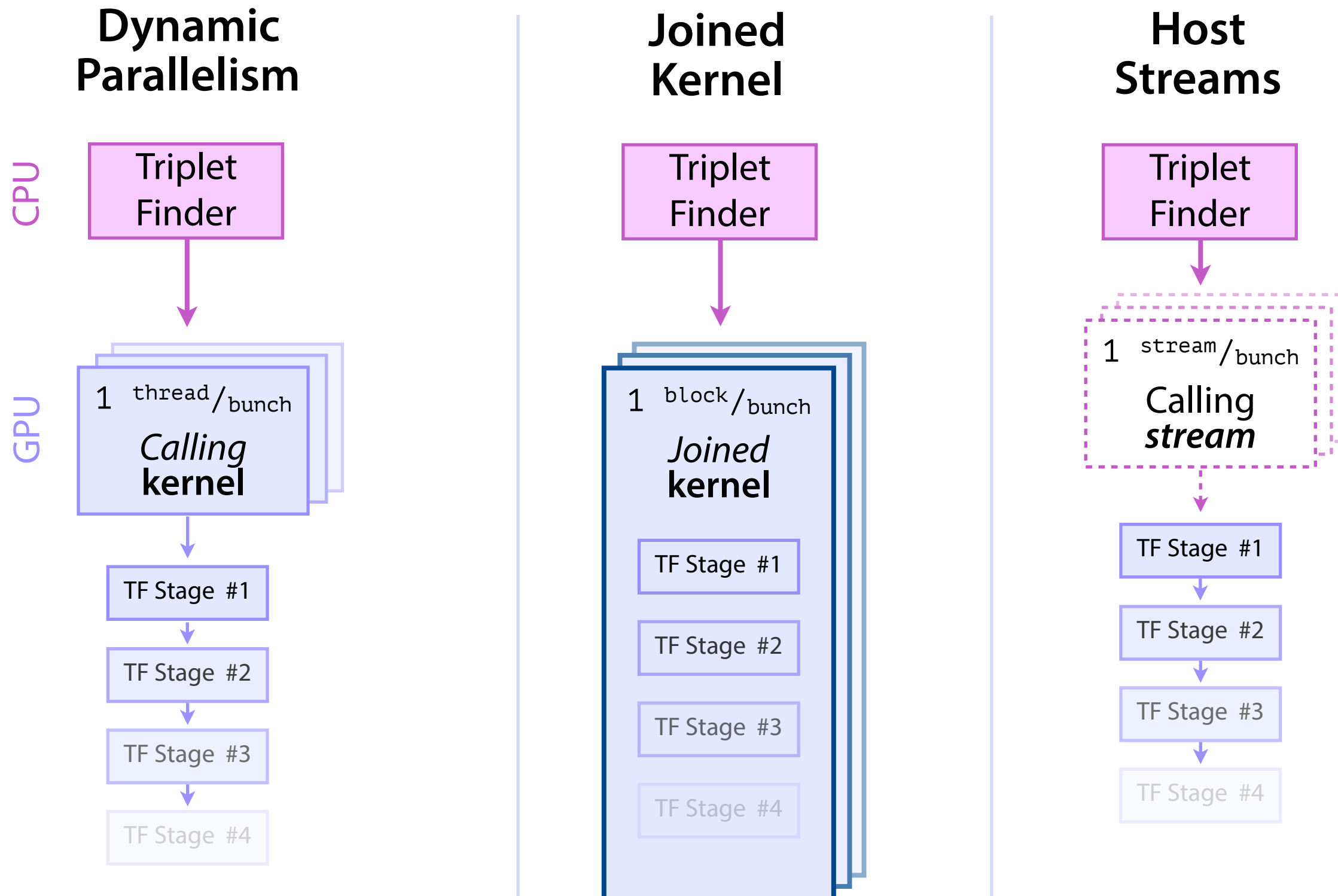
Triplet Finder — Bunching

Performance

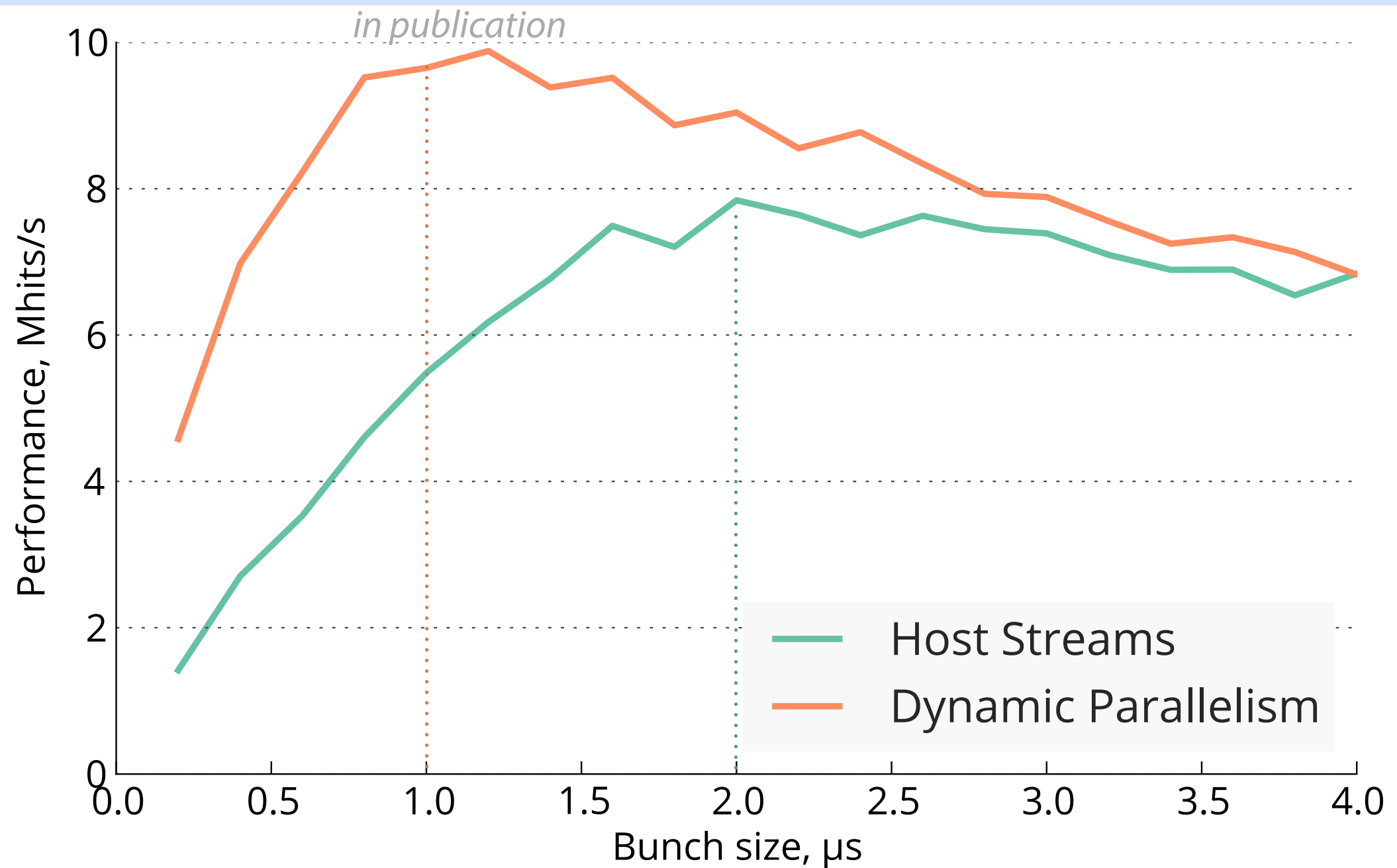


Triplet Finder — Optimizations

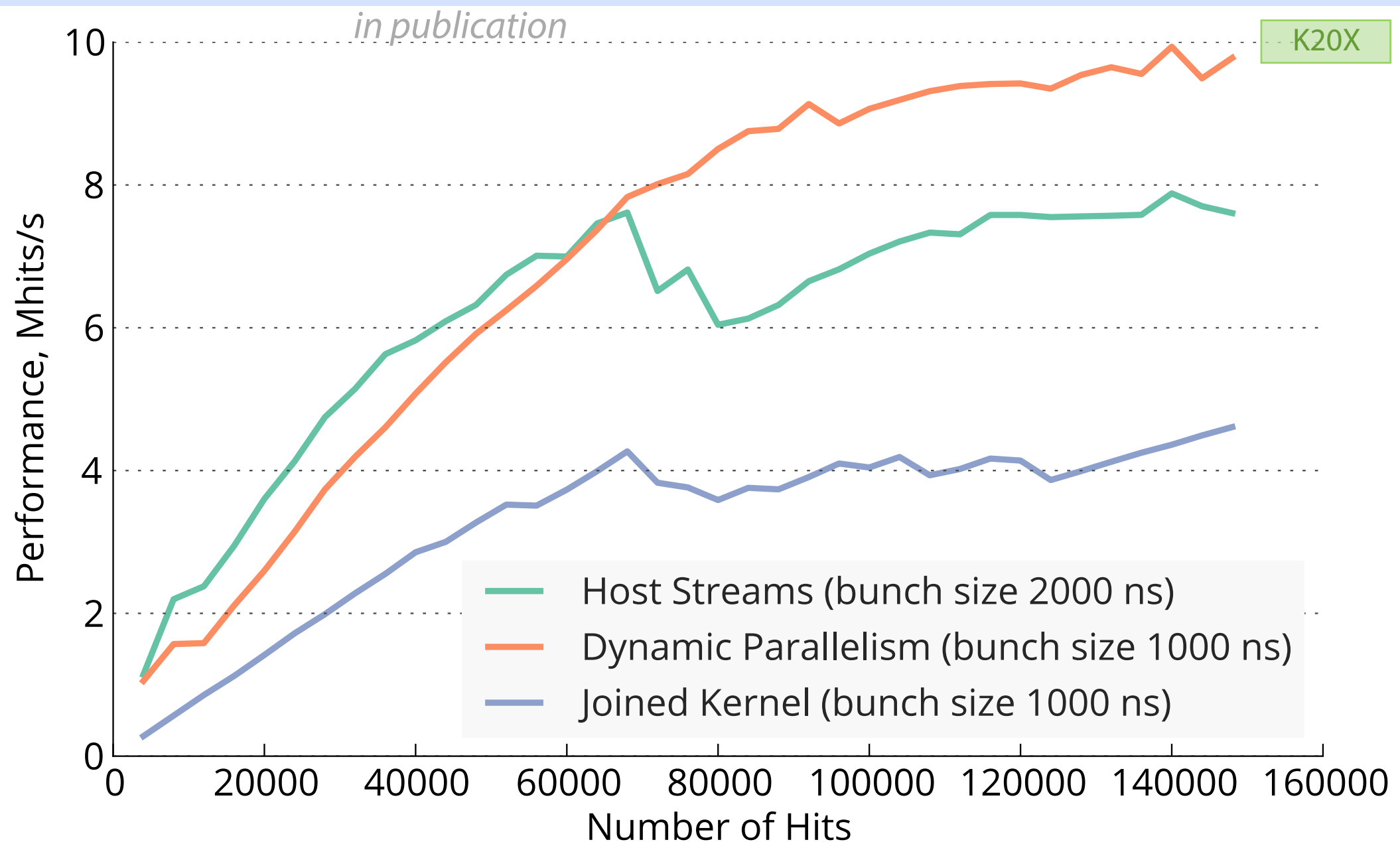
- Compare data processing strategies



Triplet Finder — Bunch Sizes

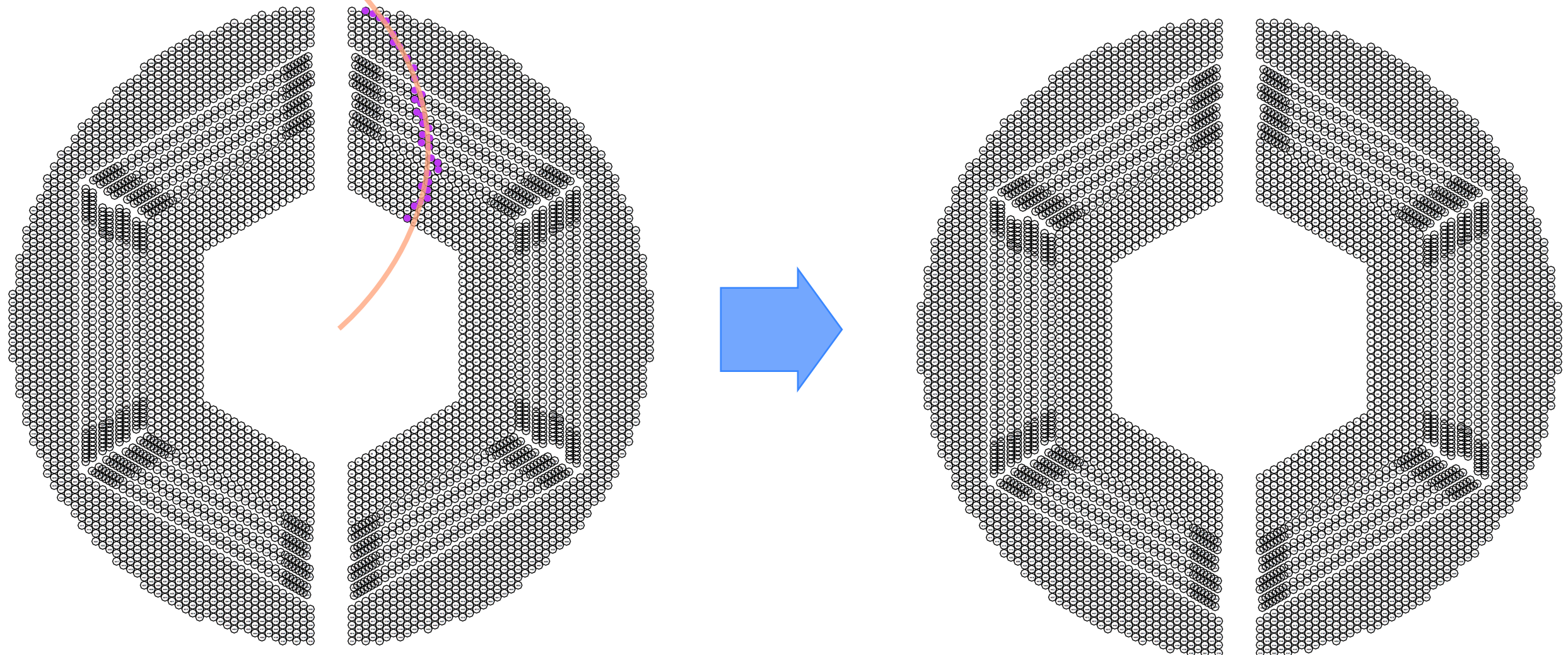


Triplet Finder — Data Processing



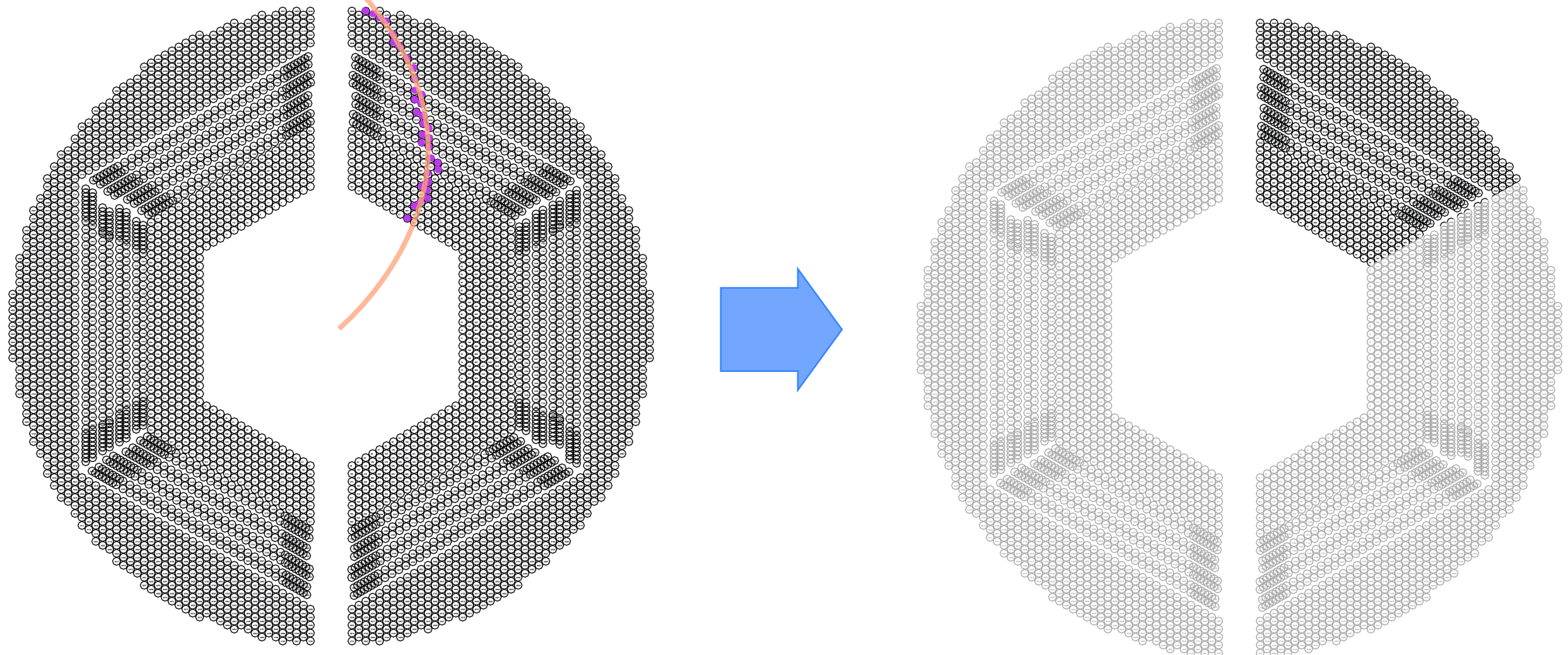
Triplet Finder — Binning: Sector Rows

- Sector Row testing
 - After found track:
Hit association not with *all* hits of current window,
but only with subset
(*first test rows of sector, then hits of row*)



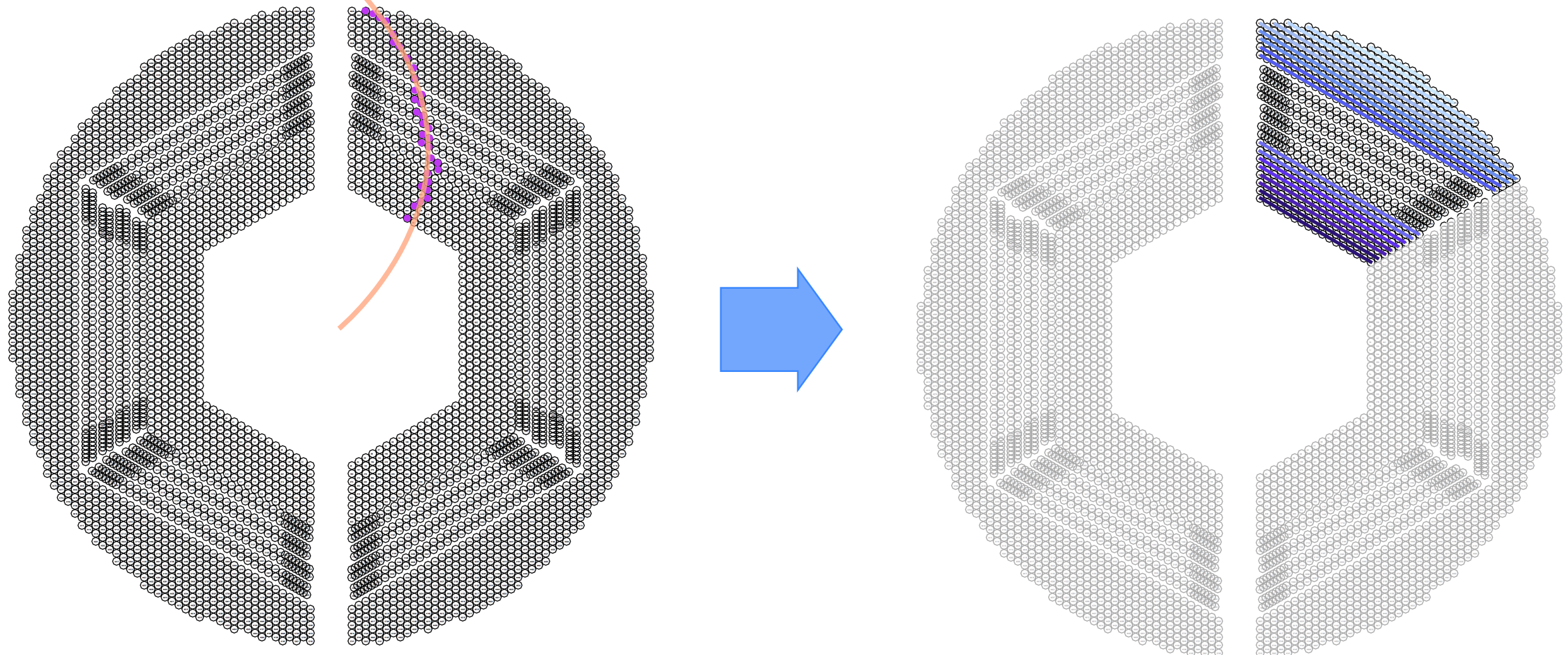
Triplet Finder — Binning: Sector Rows

- Sector Row testing
 - After found track:
Hit association not with *all* hits of current window,
but only with subset
(*first test rows of sector, then hits of row*)



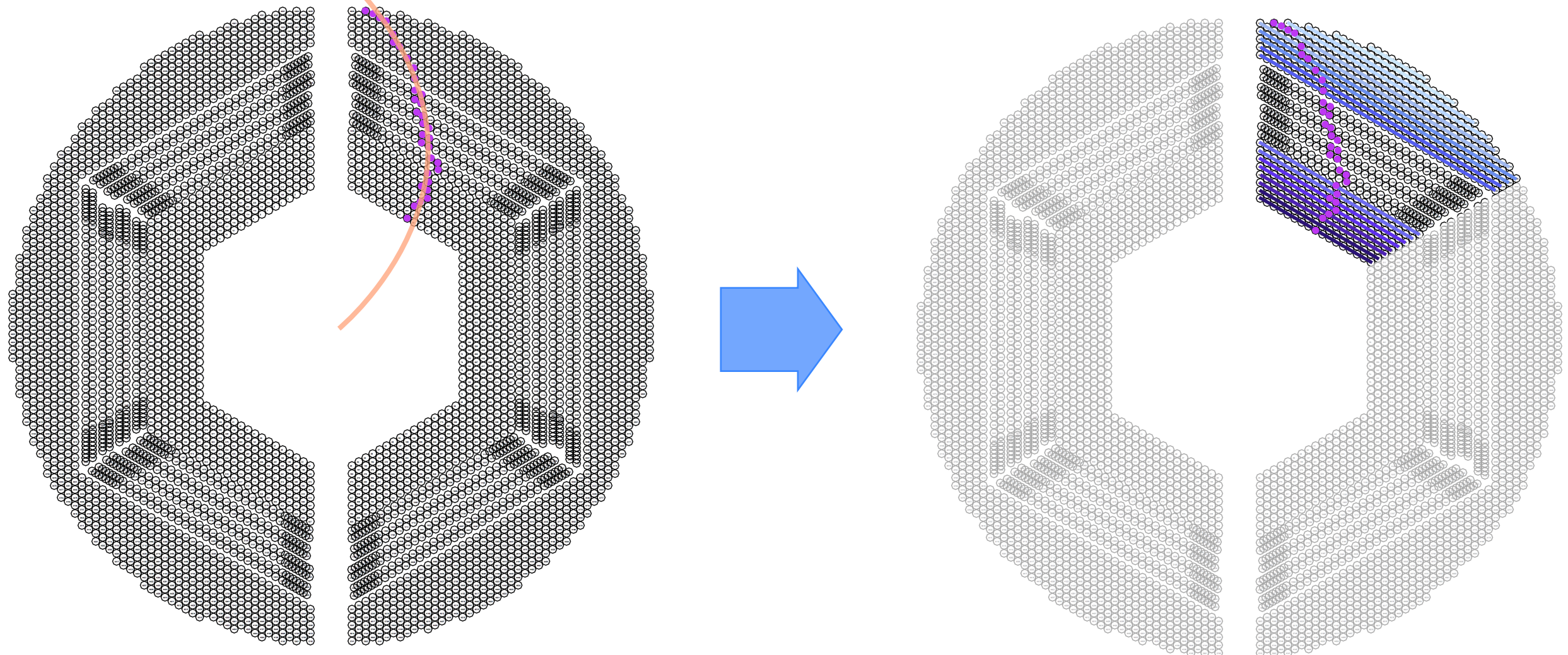
Triplet Finder — Binning: Sector Rows

- Sector Row testing
 - After found track:
Hit association not with *all* hits of current window,
but only with subset
(*first test rows of sector, then hits of row*)



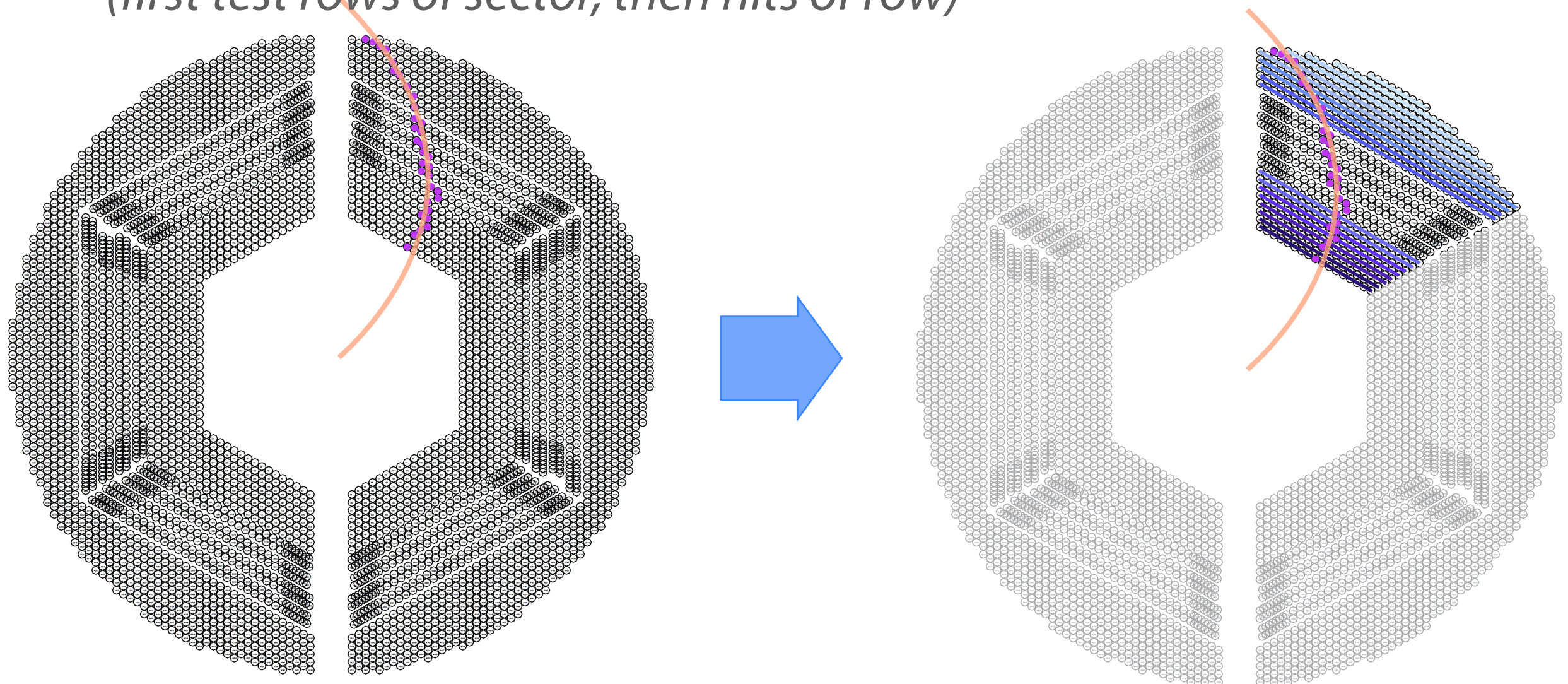
Triplet Finder — Binning: Sector Rows

- Sector Row testing
 - After found track:
Hit association not with *all* hits of current window,
but only with subset
(*first test rows of sector, then hits of row*)

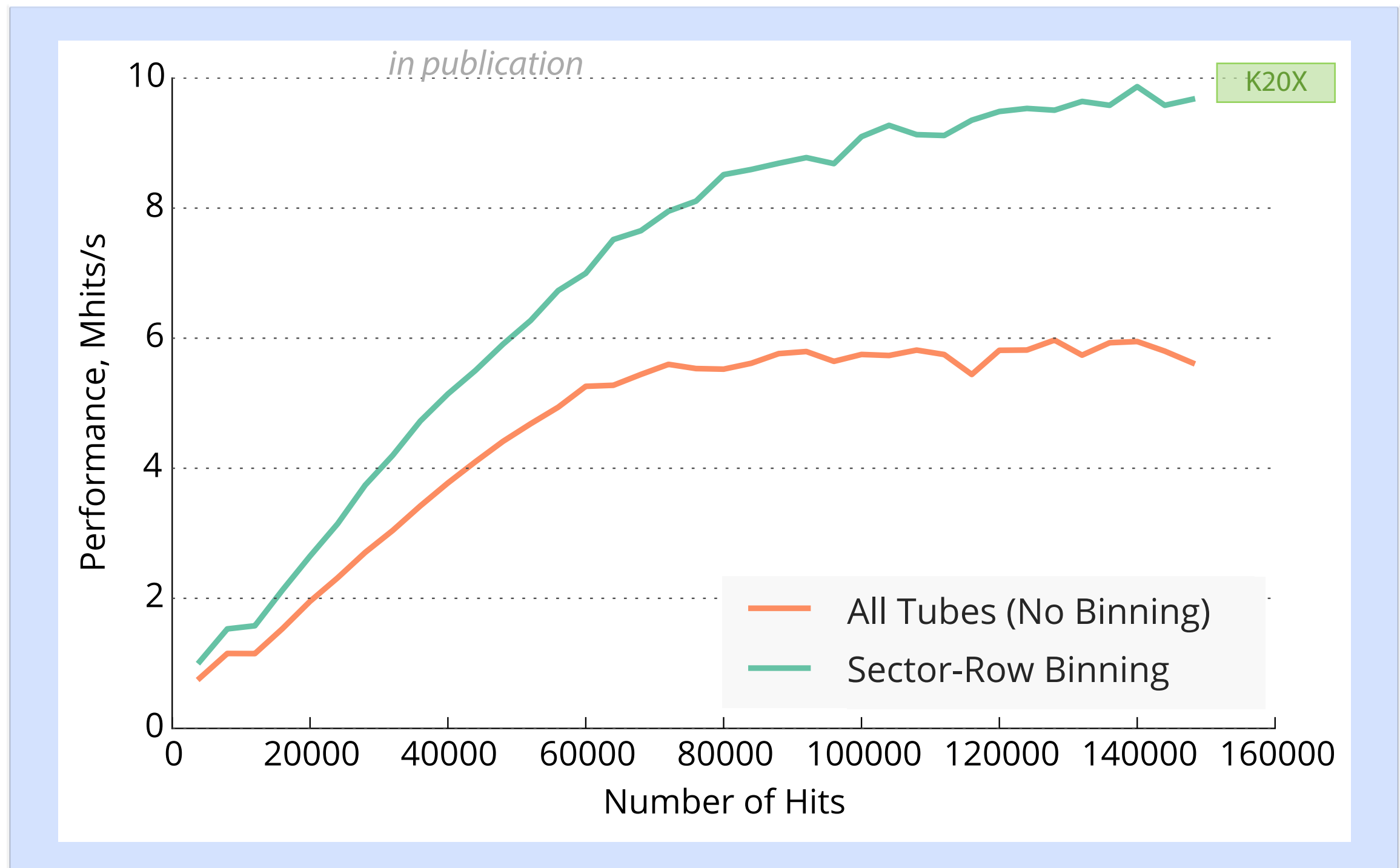


Triplet Finder — Binning: Sector Rows

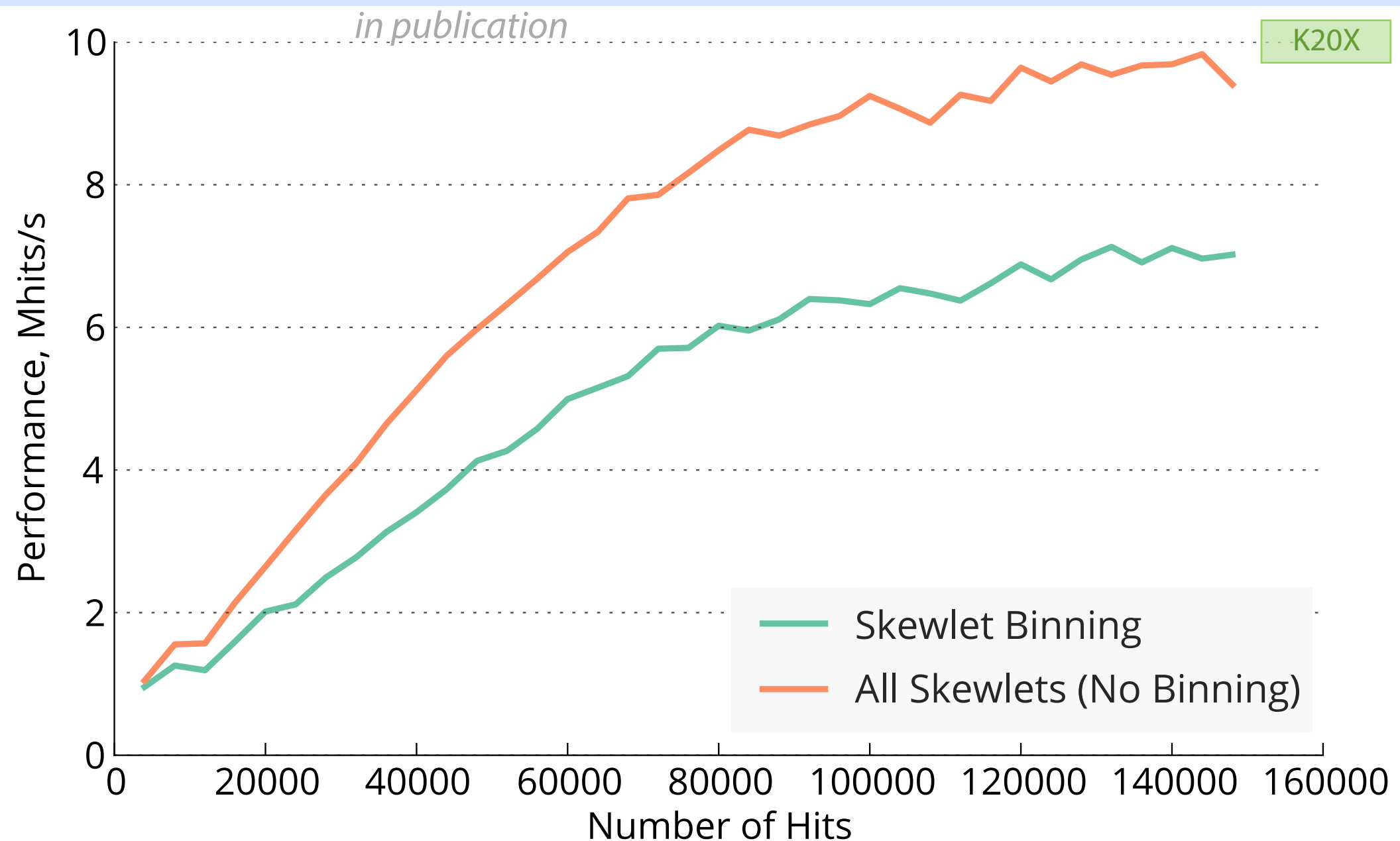
- Sector Row testing
 - After found track:
Hit association not with *all* hits of current window,
but only with subset
(*first test rows of sector, then hits of row*)



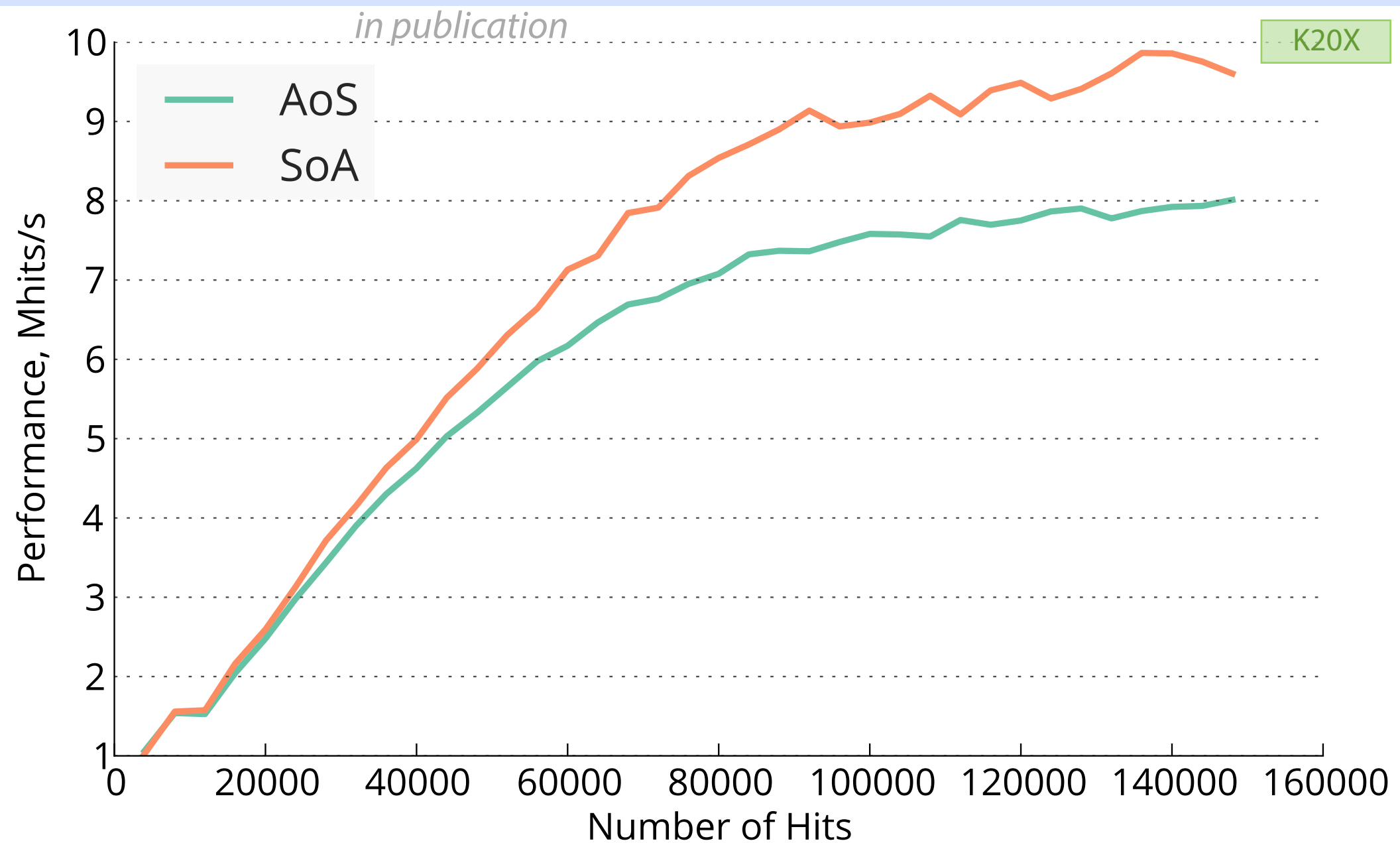
Triplet Finder — Binning: Sector Rows



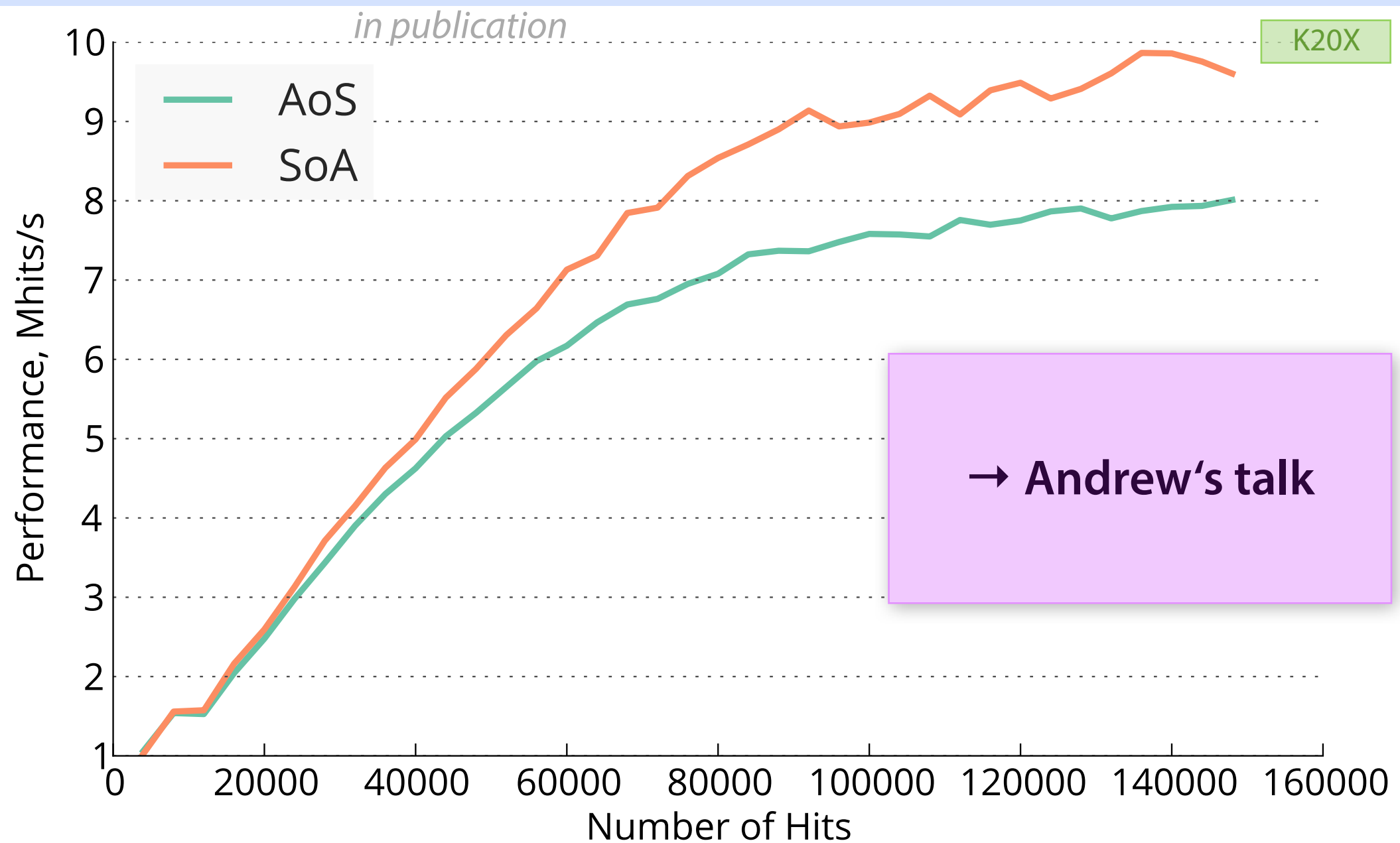
Triplet Finder — Binning: Skewlets



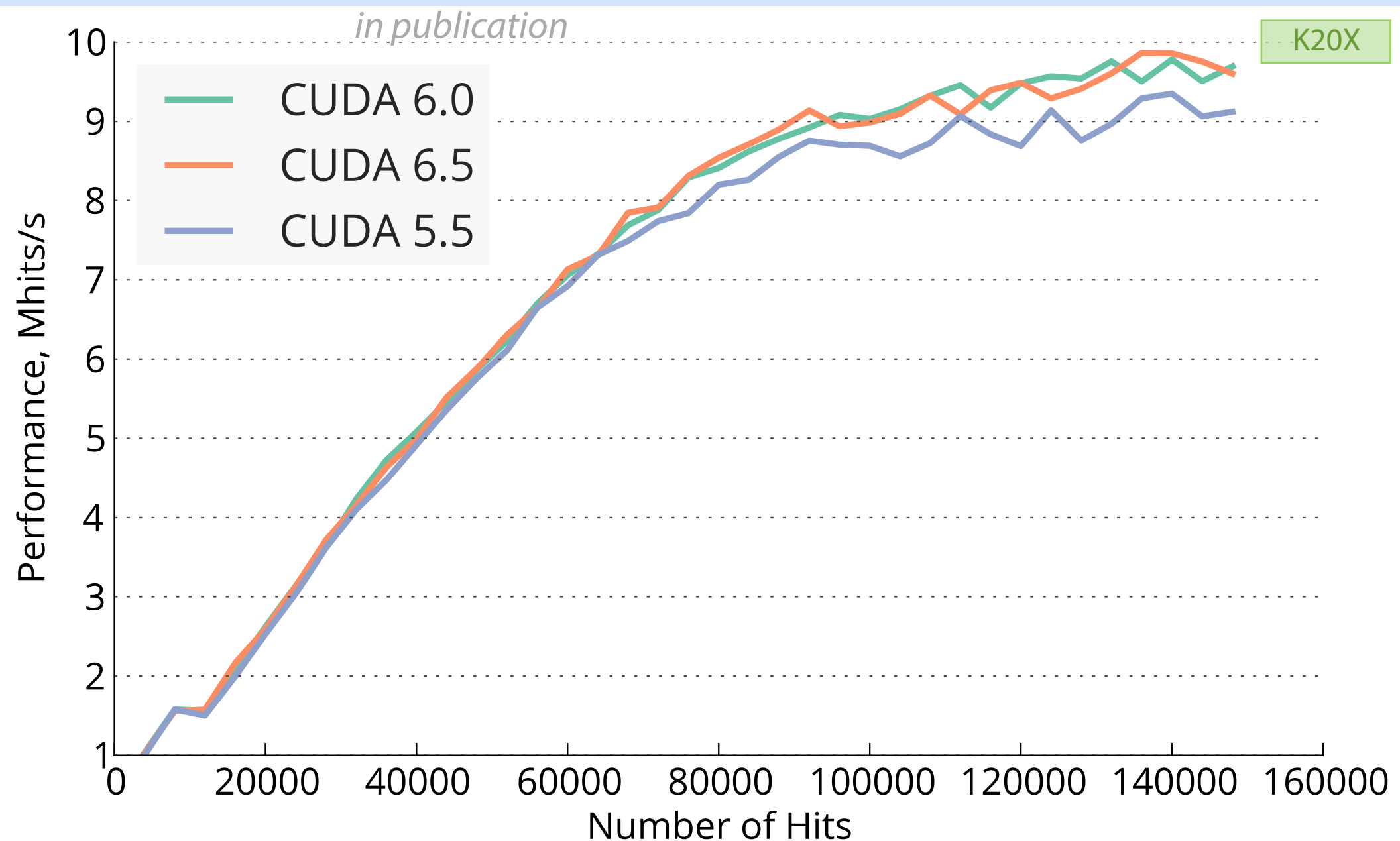
Triplet Finder — AoS vs. SoA



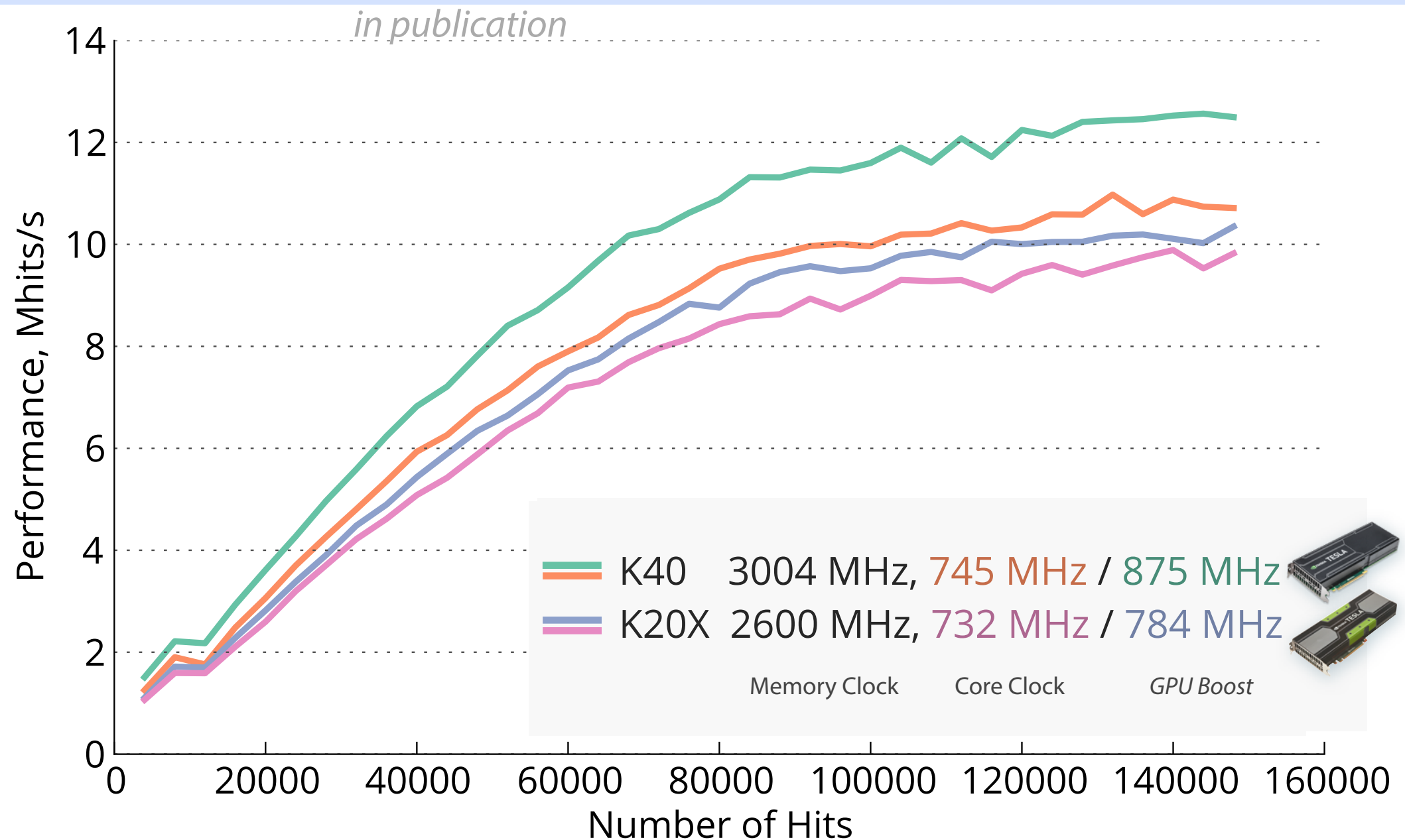
Triplet Finder — AoS vs. SoA



Triplet Finder — CUDA Versions



Triplet Finder — Clock Speed / GPU



Triplet Finder — Summary

- Best performance: **14 μ s/event**
 - $14 \cdot 10^{-6} \text{ s/event} * 2 \cdot 10^7 \text{ event/s} \Rightarrow 280 \text{ GPUs}^{2014}$
 - PANDA²⁰¹⁹: Multi GPU system – $\mathcal{O}(100)$ GPUs
- Optimizations possible & needed
 - ε needs to be improved
 - Speed, €:
 - *More float less double*-cards a la K10
 - Consumer-grade cards a la GTX

- PANDA researches in using GPUs as part of online event reconstruction scheme
- Algorithms in active evaluation and optimization
 - Triplet Finder performance-optimized
- Data transfer to GPU in research: FairMQ

- PANDA researches in using GPUs as part of online event reconstruction scheme
- Algorithms in active evaluation and optimization
 - Triplet Finder performance-optimized
- Data transfer to GPU in research: FairMQ

Thank you!

Andreas Herten
a.herten@fz-juelich.de

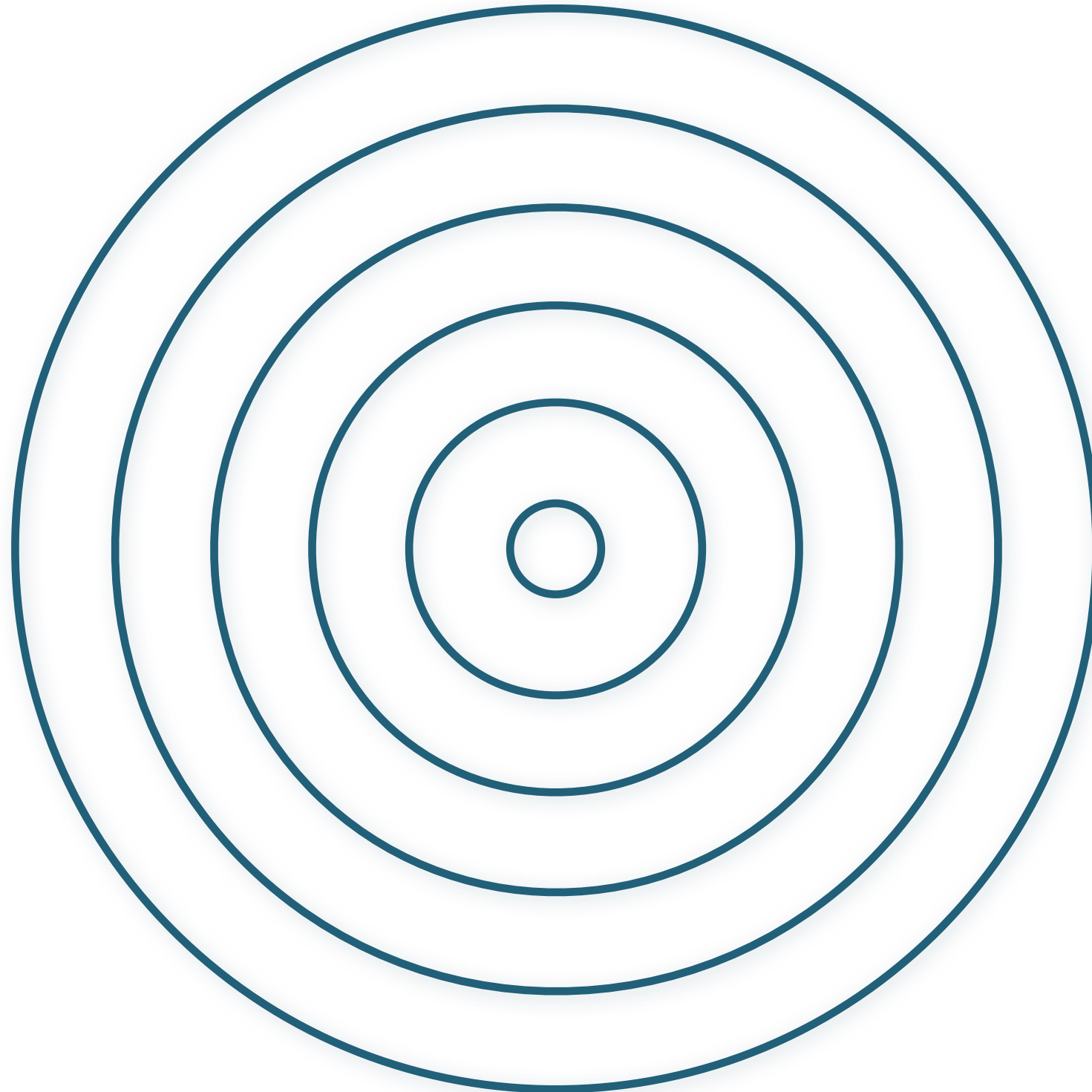


List of Resources Used

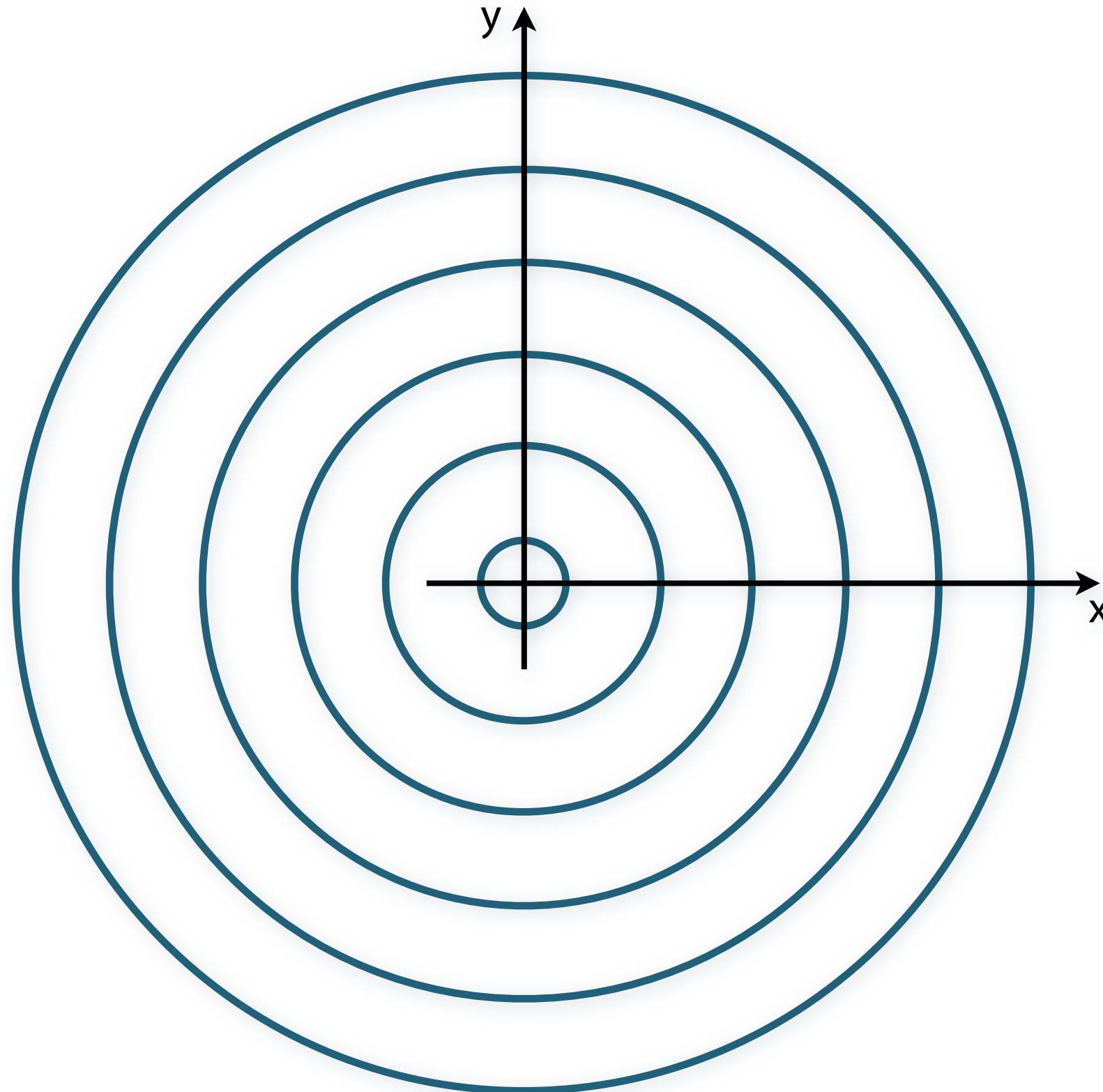
- **#3:** Jupiter icon by Nikki Rodriguez from The Noun Project
- **#3:** Einstein icon by Roman Rusinov from The Noun Project
- **#6:** FAIR vector logo from official FAIR website
- **#6:** FAIR rendering from official website
- **#11:** Flare Gun icon by Jop van der Kroef from The Noun Project
- **#27:** STT event animation by Marius C. Mertens
- **#35:** Graphics cards images by NVIDIA promotion
- **#35:** GPU Specifications
 - Tesla K20X Specifications: <http://www.nvidia.com/content/PDF/kepler/Tesla-K20X-BD-06397-001-v07.pdf>
 - Tesla K40 Specifications: http://www.nvidia.com/content/PDF/kepler/Tesla-K40-Active-Board-Spec-BD-06949-001_v03.pdf
 - Tesla Family Overview: <http://www.nvidia.com/content/tesla/pdf/NVIDIA-Tesla-Kepler-Family-Datasheet.pdf>

BACKUP

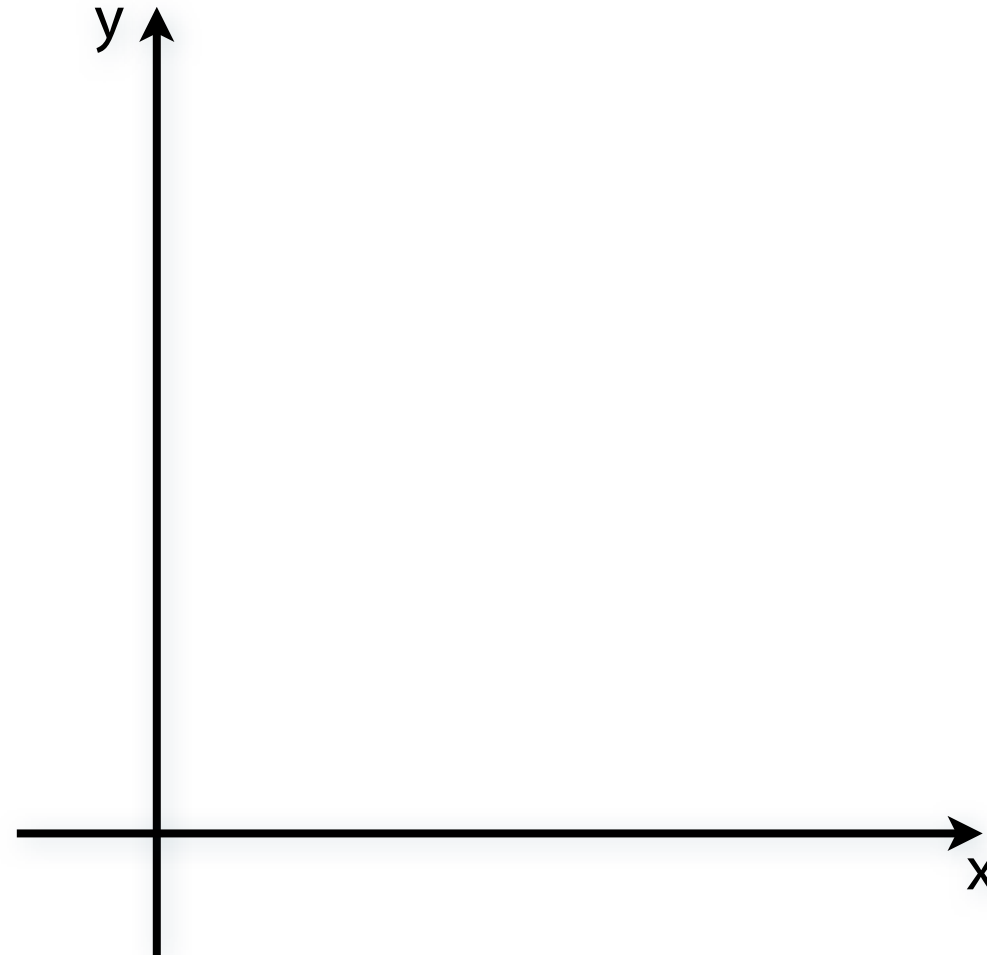
Hough Transform — Principle



Hough Transform — Principle

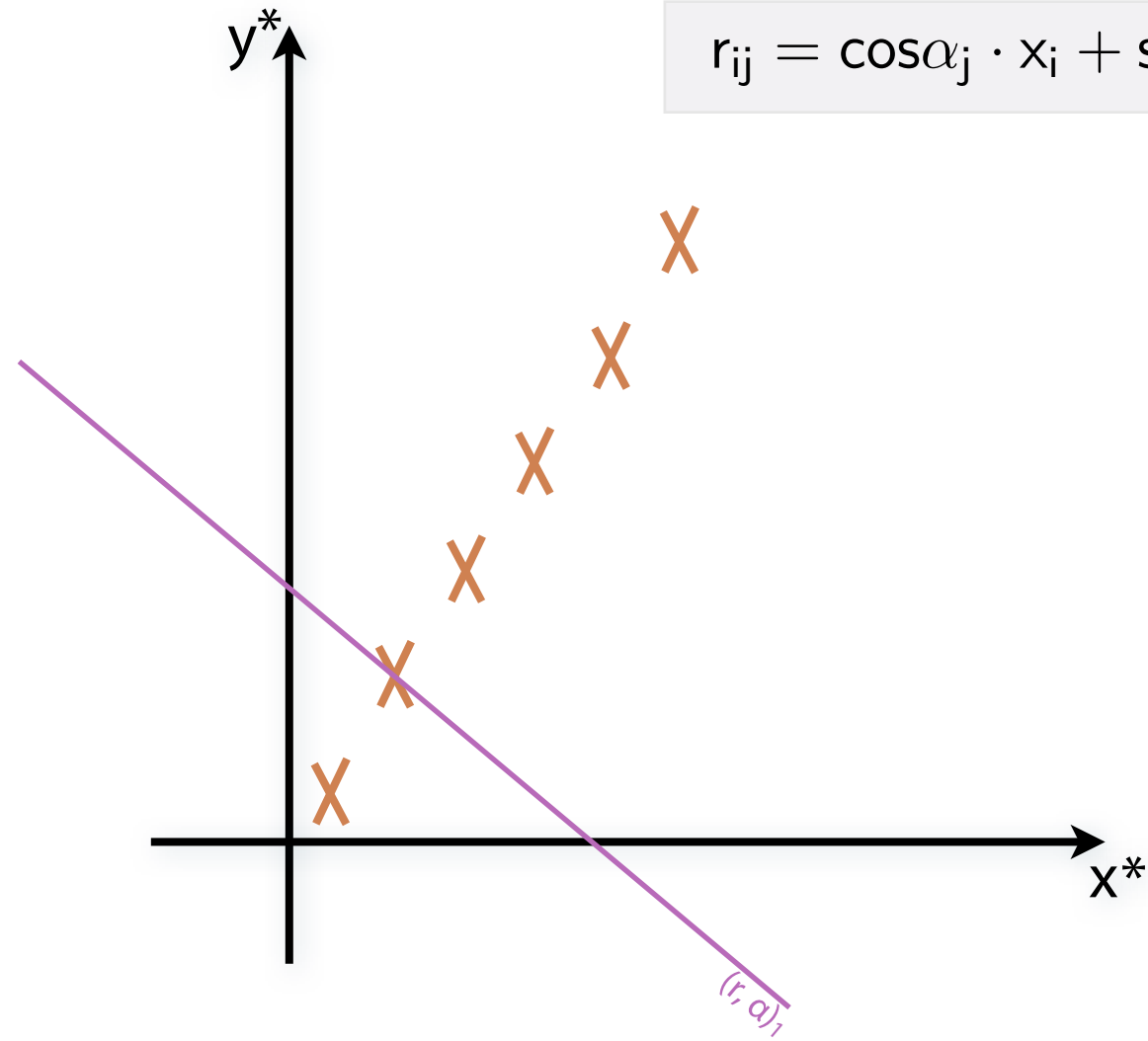


Hough Transform — Principle



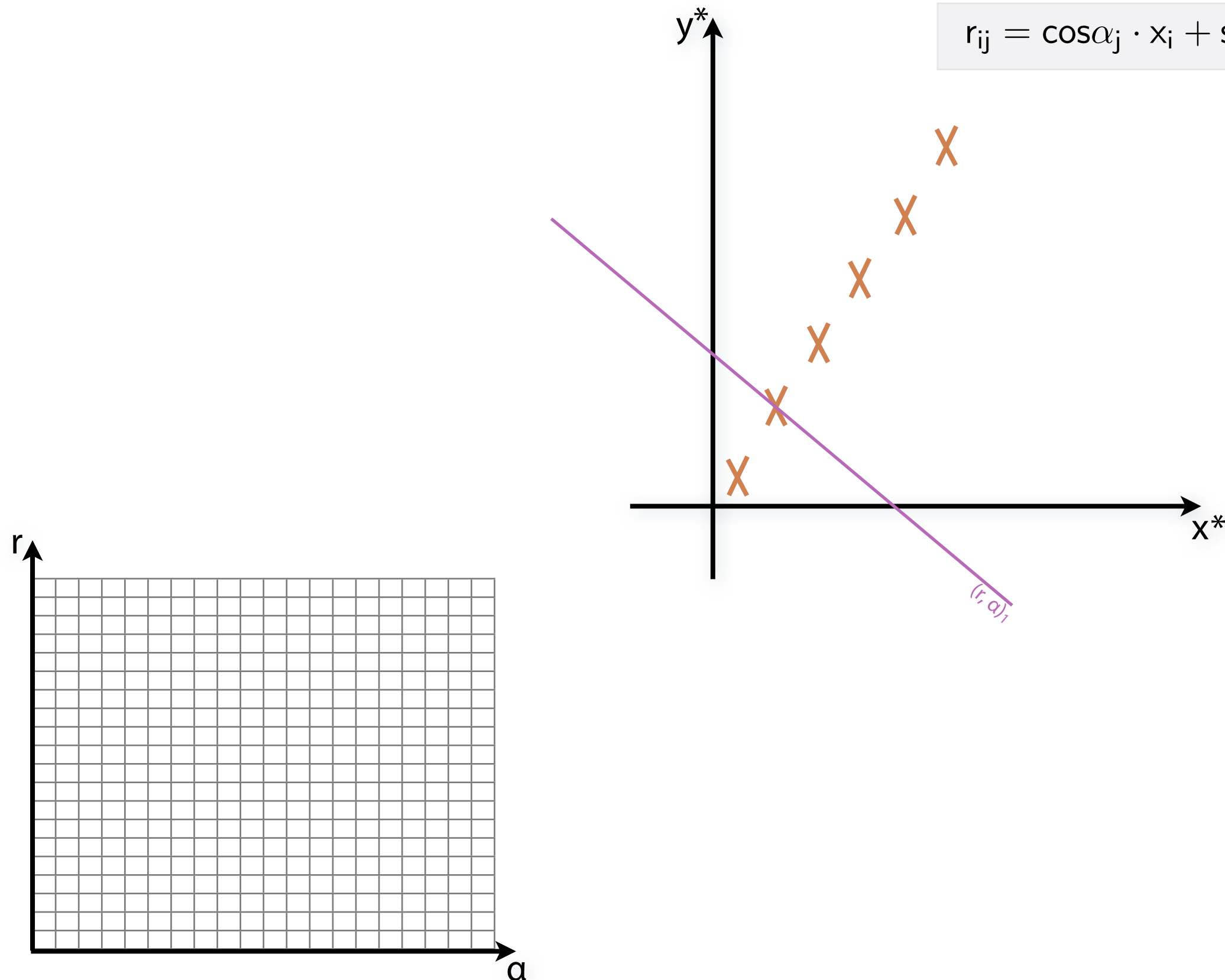
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



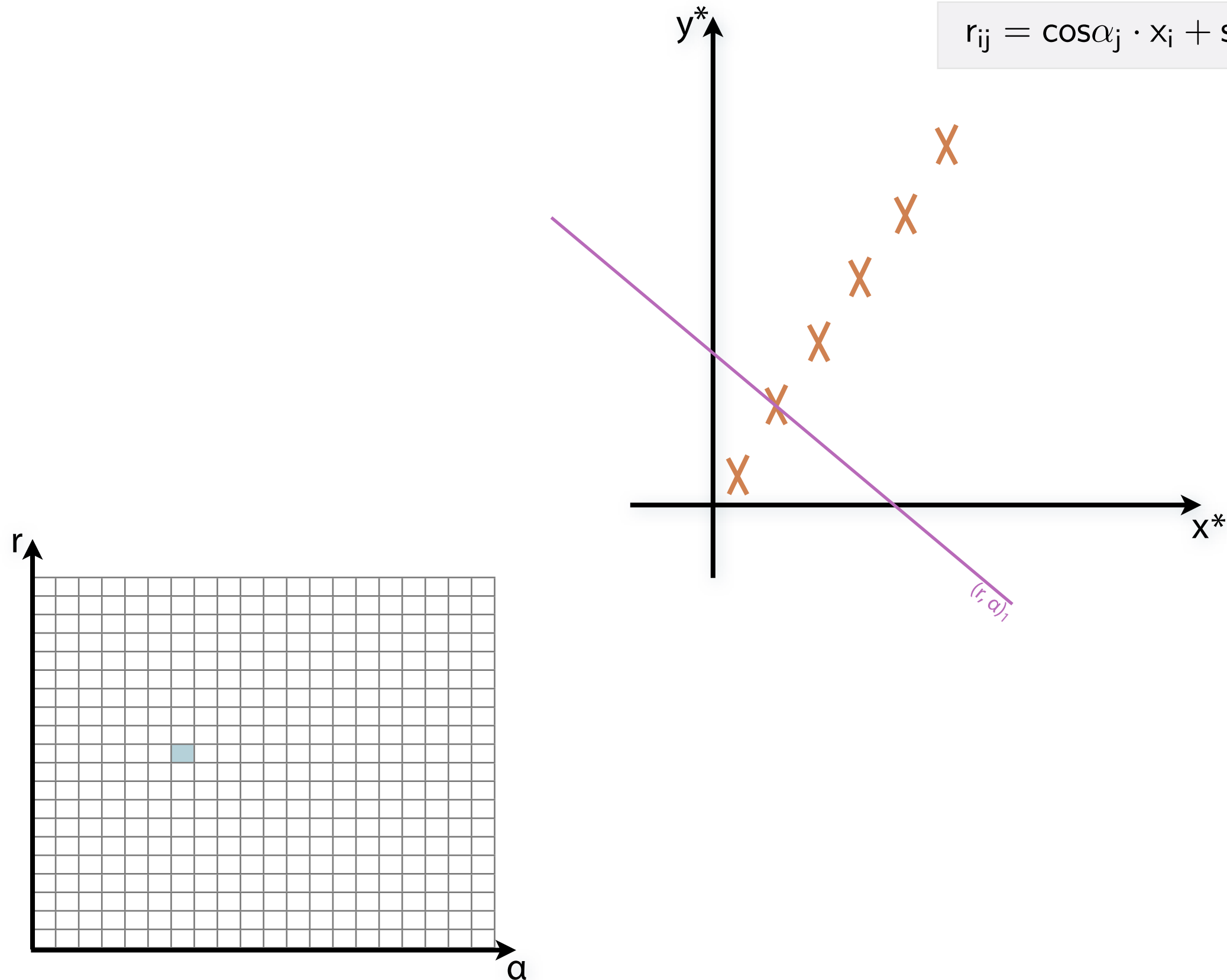
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



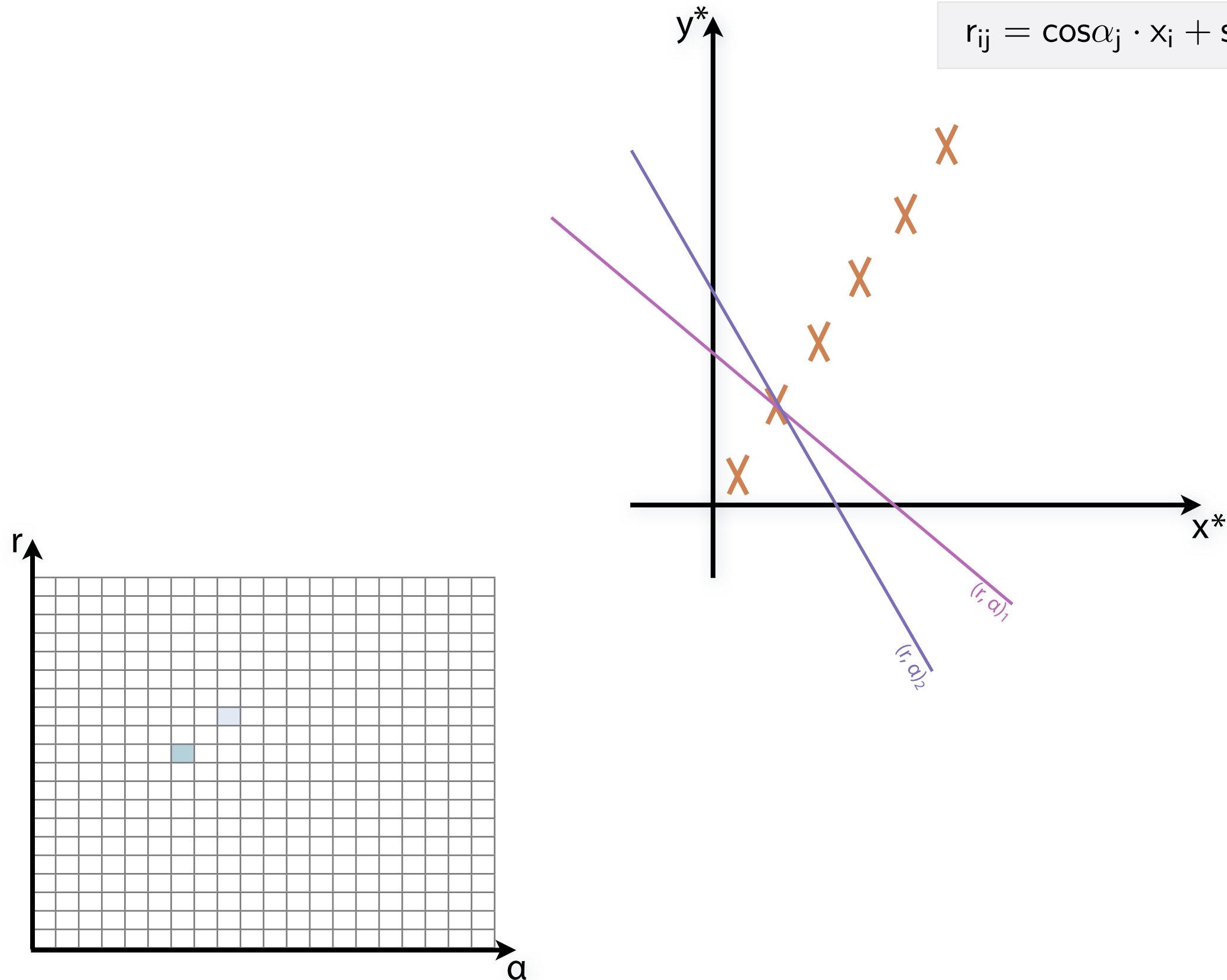
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



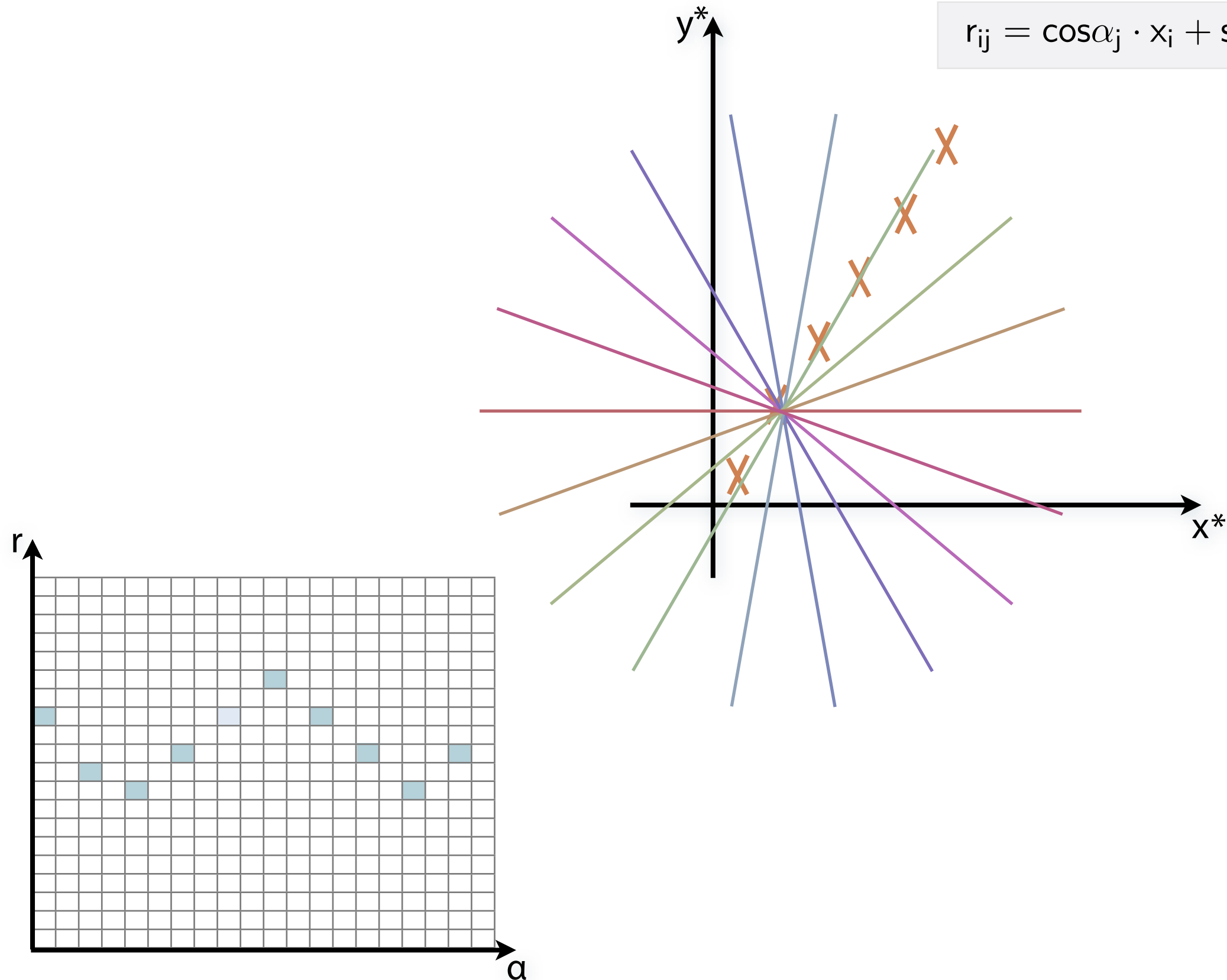
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



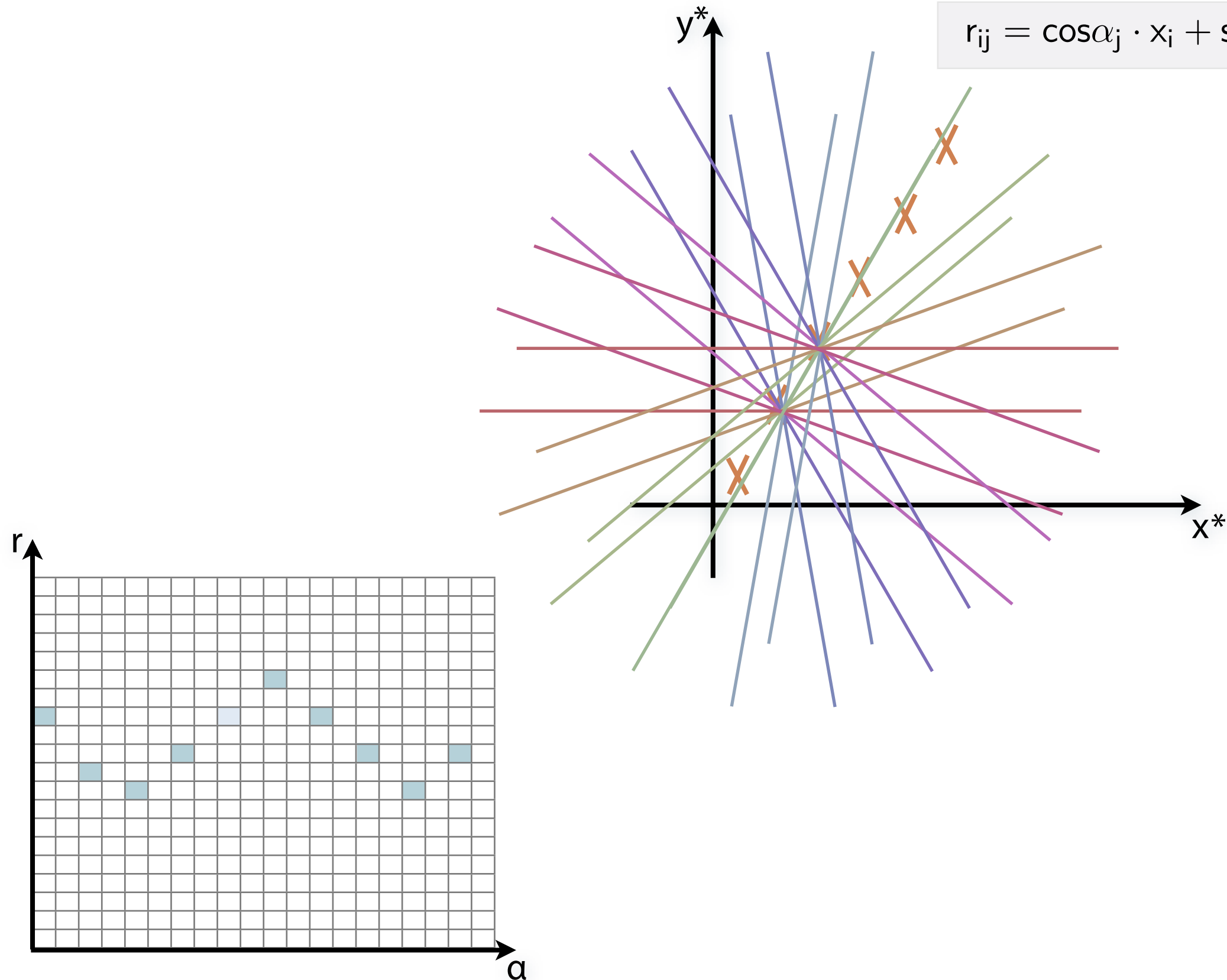
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



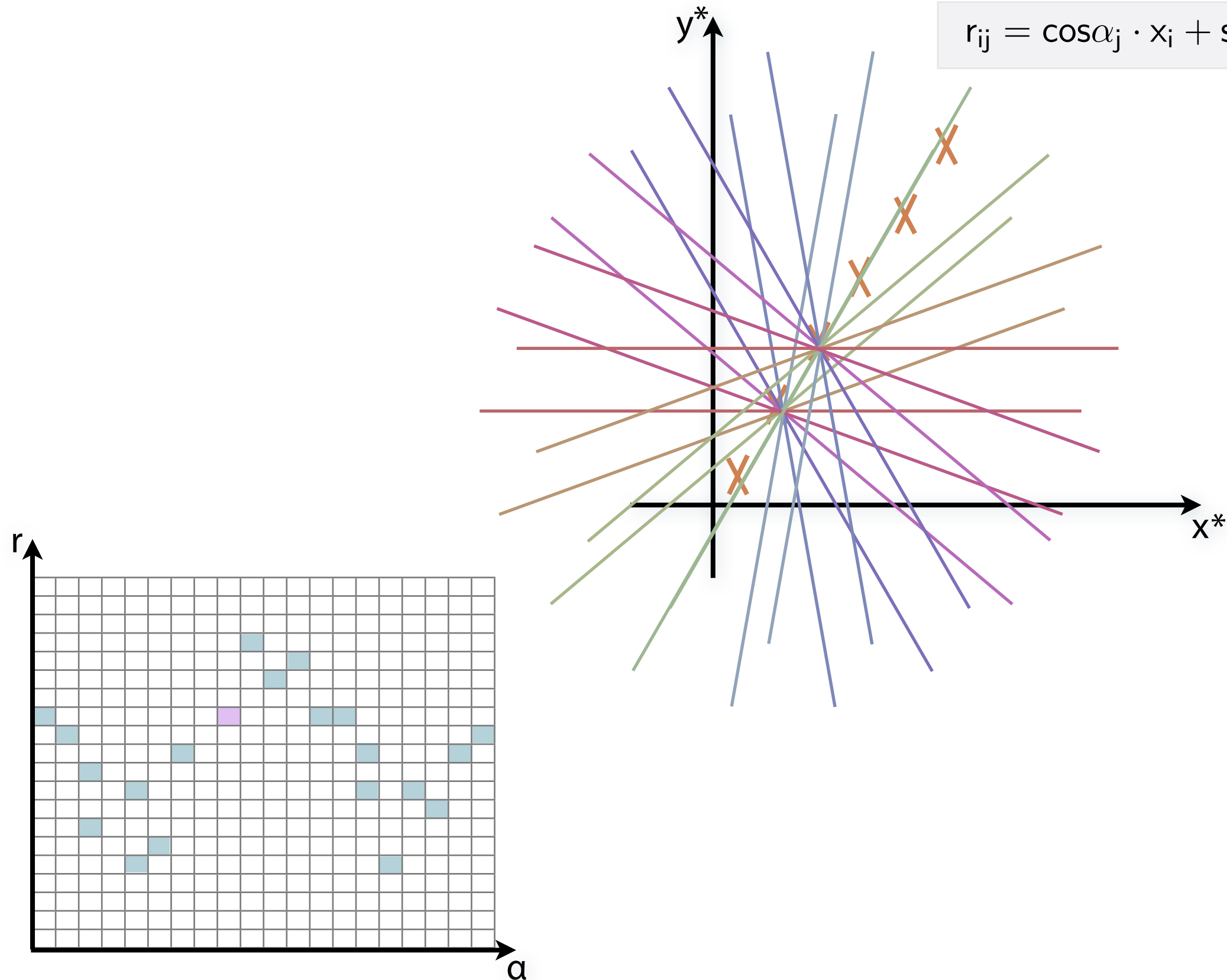
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



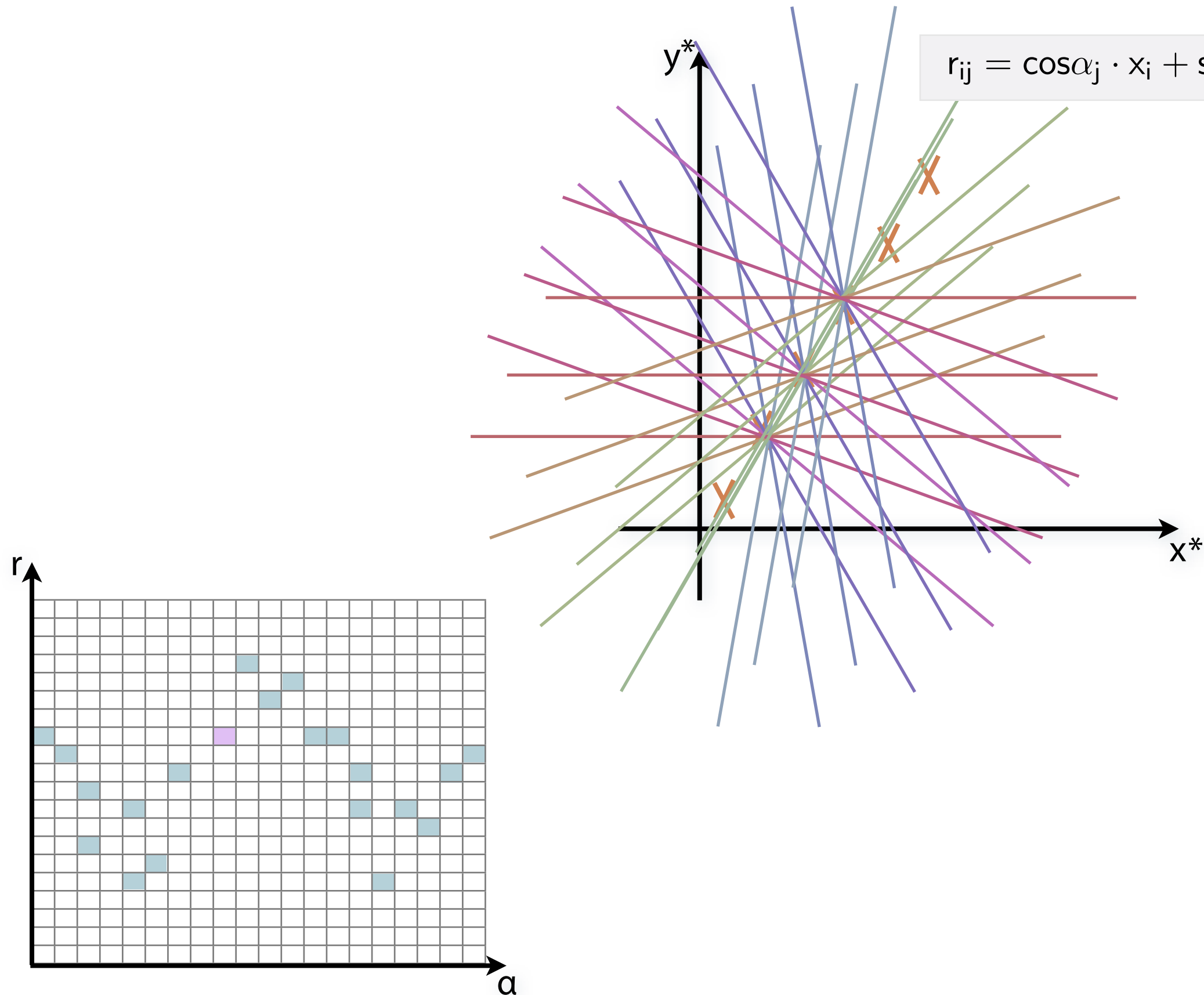
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



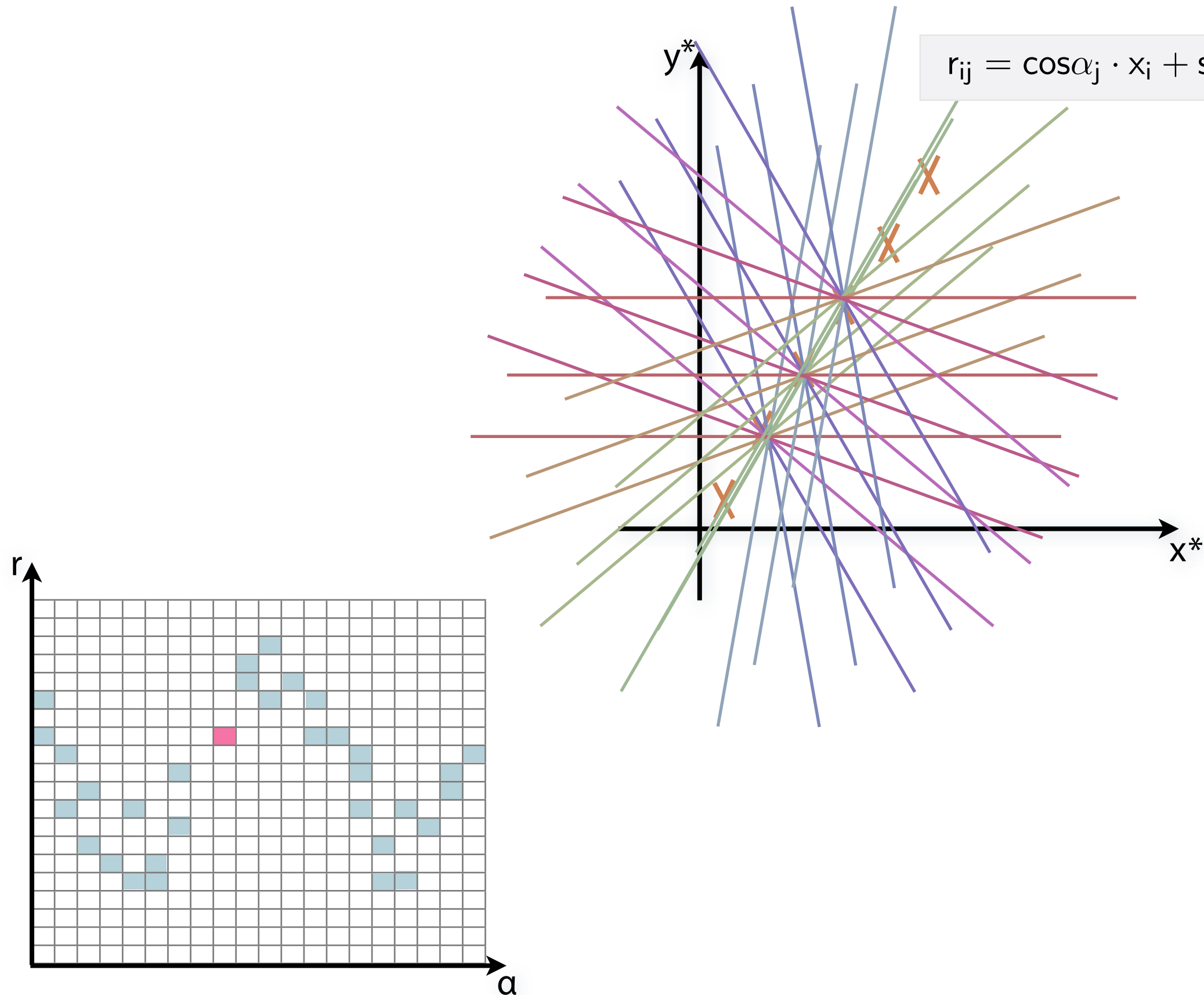
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



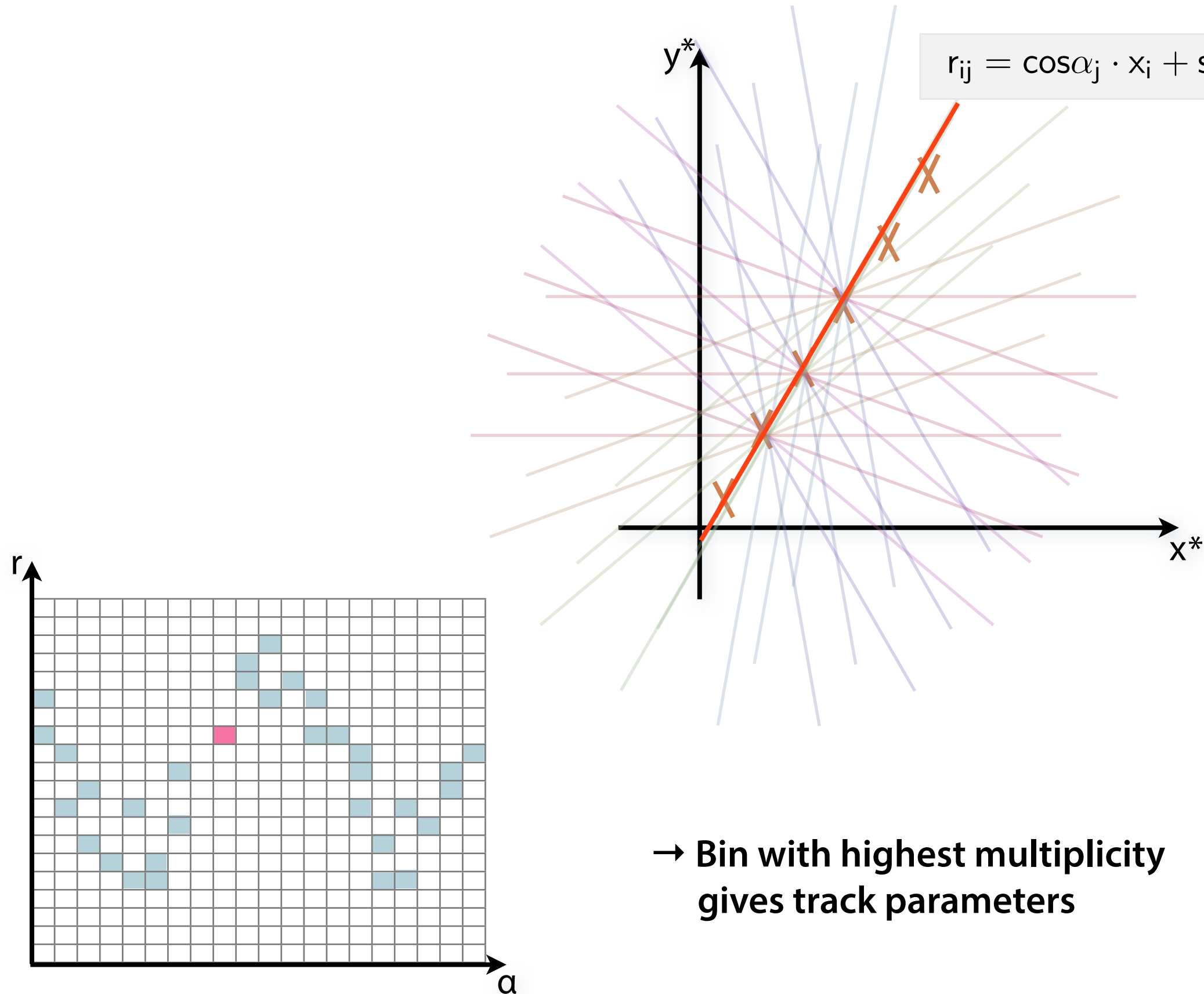
Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



Hough Transform — Principle

$$r_{ij} = \cos\alpha_j \cdot x_i + \sin\alpha_j \cdot y_i + \rho_i$$



Riemann Algorithm — Procedure

- 1 **Create triplet of hit points**
 - All possible three hit combinations need to become triplets

1 Create triplet of hit points

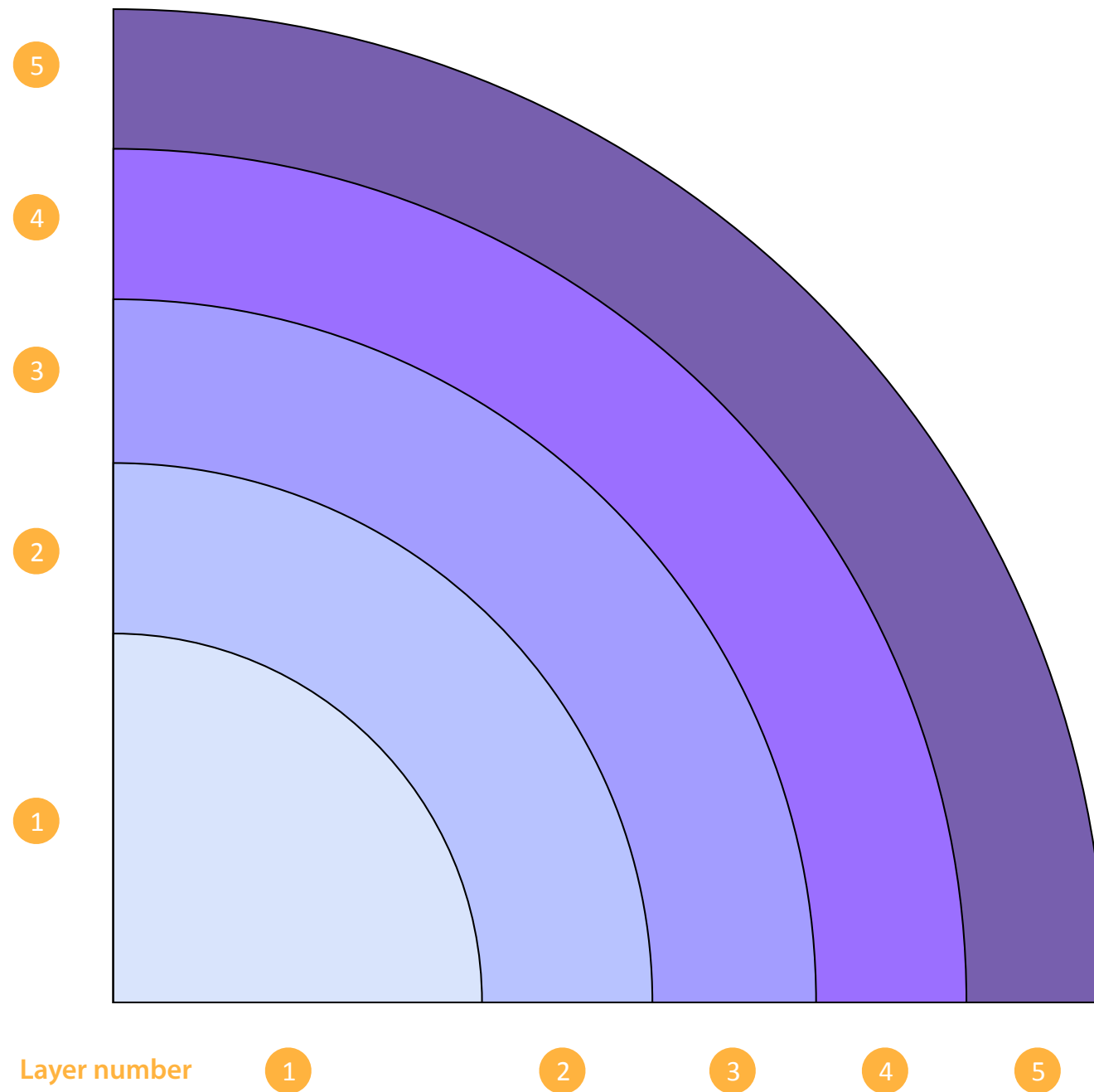
- All possible three hit combinations need to become triplets

2 Grow triplets to tracks:

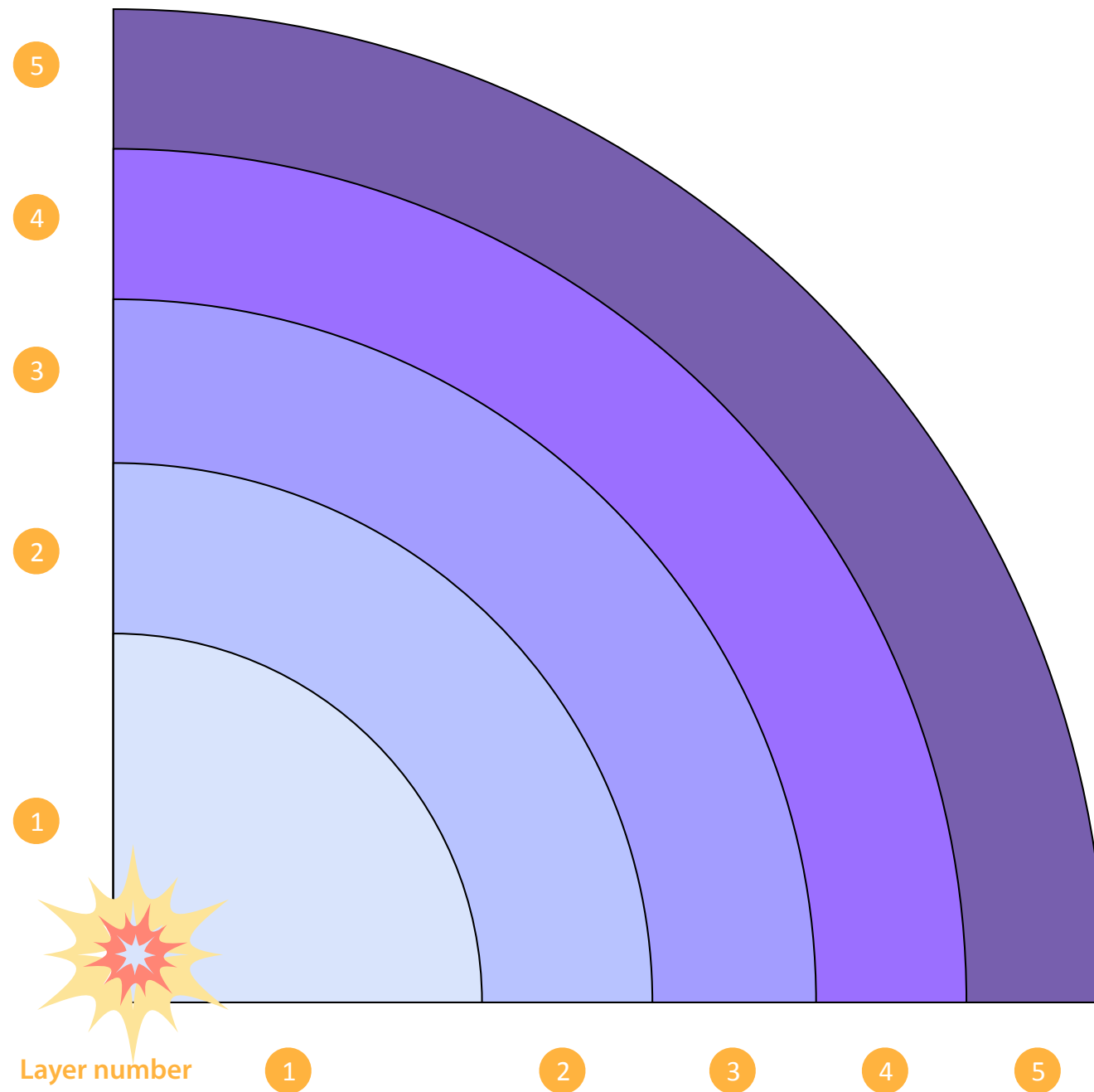
Continuously test next hit if it fits to triplet track

- Use Riemann paraboloid to circle fit track
 - Test closeness of new hit: good → add hit; bad → dismiss hit
 - Continue with next hit
- Helix fit: arc length s vs. z position

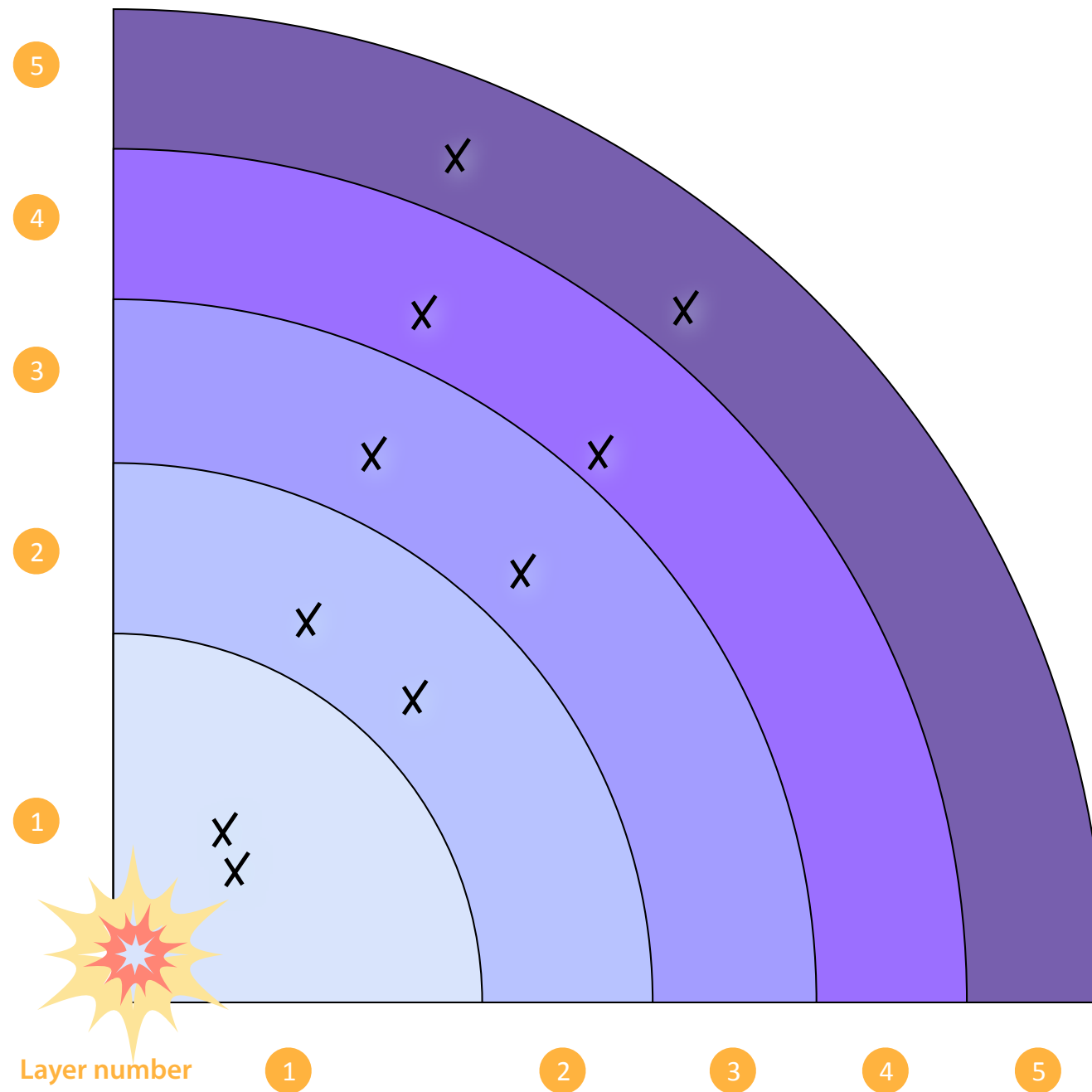
Riemann Track Finder — 1 Seeds



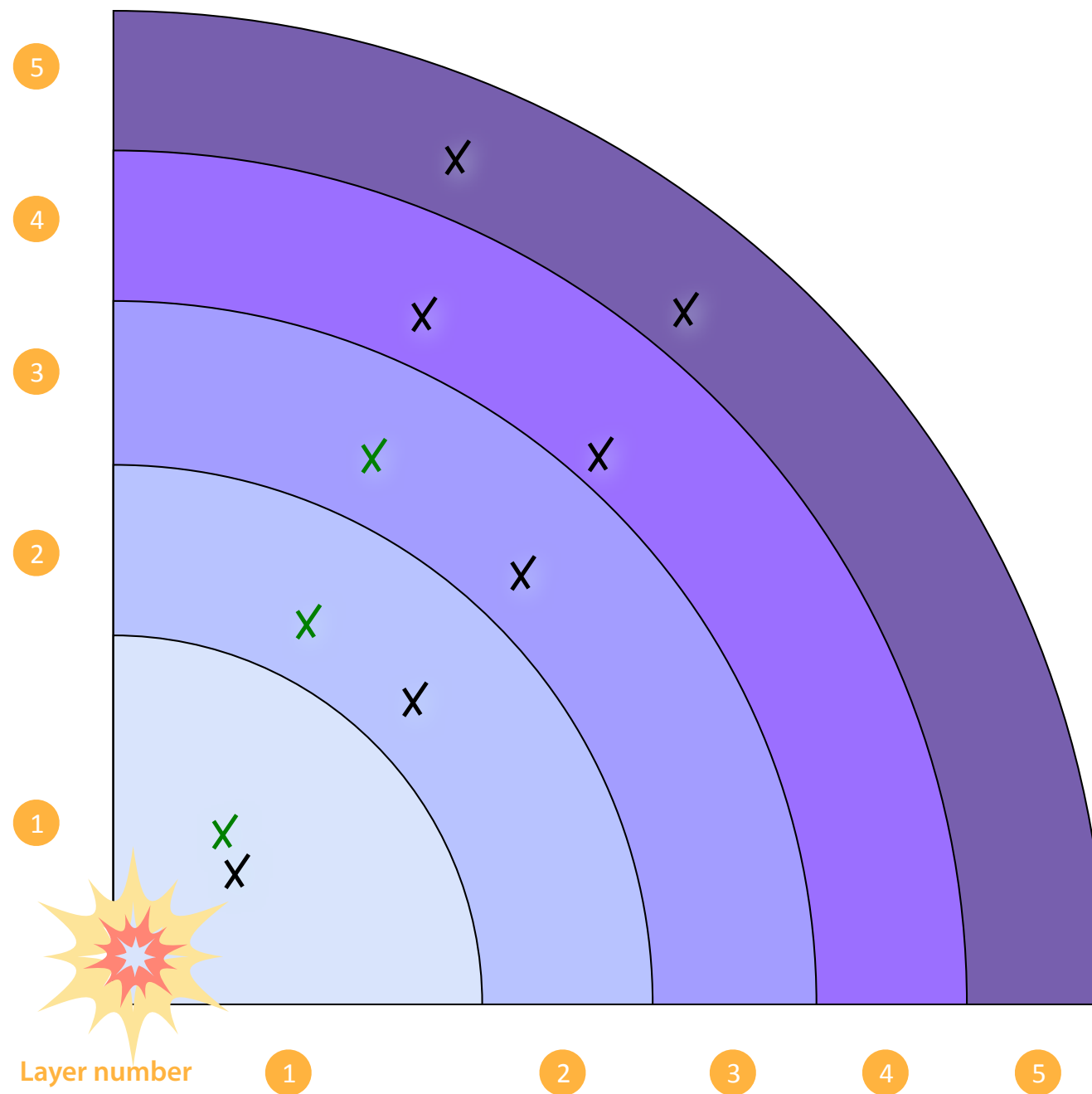
Riemann Track Finder — 1 Seeds



Riemann Track Finder — 1 Seeds

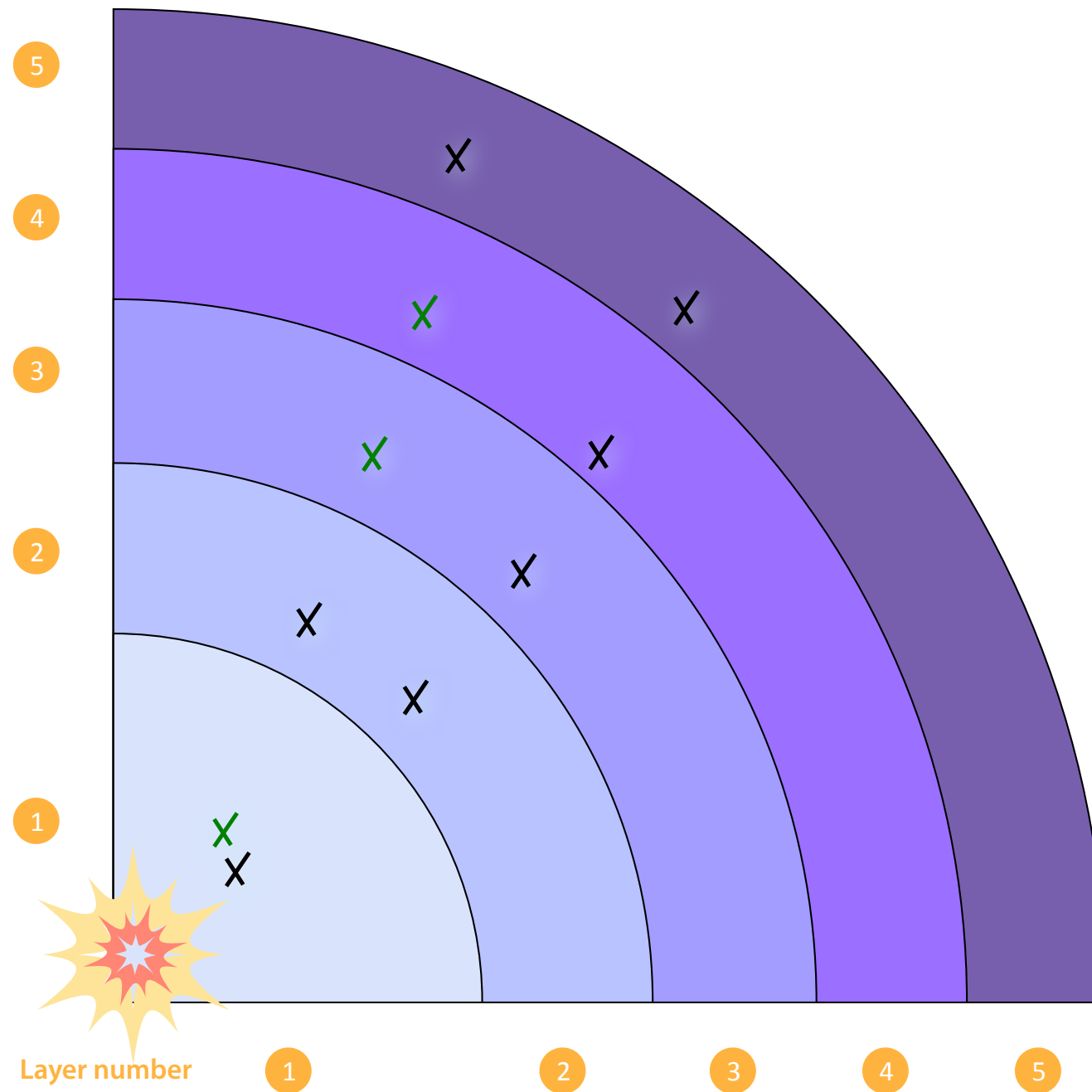


Riemann Track Finder — 1 Seeds

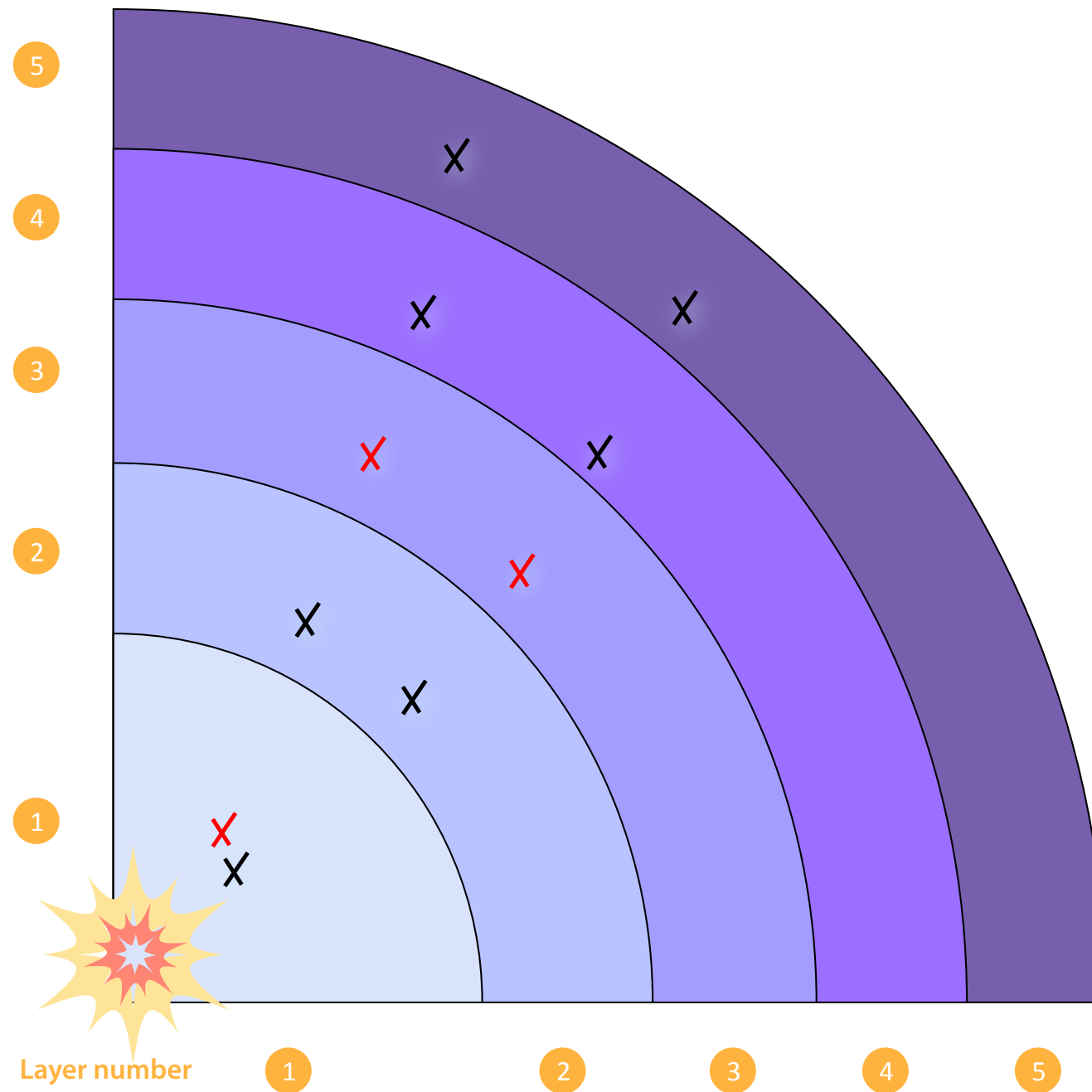


1₁ 2₁ 3₁

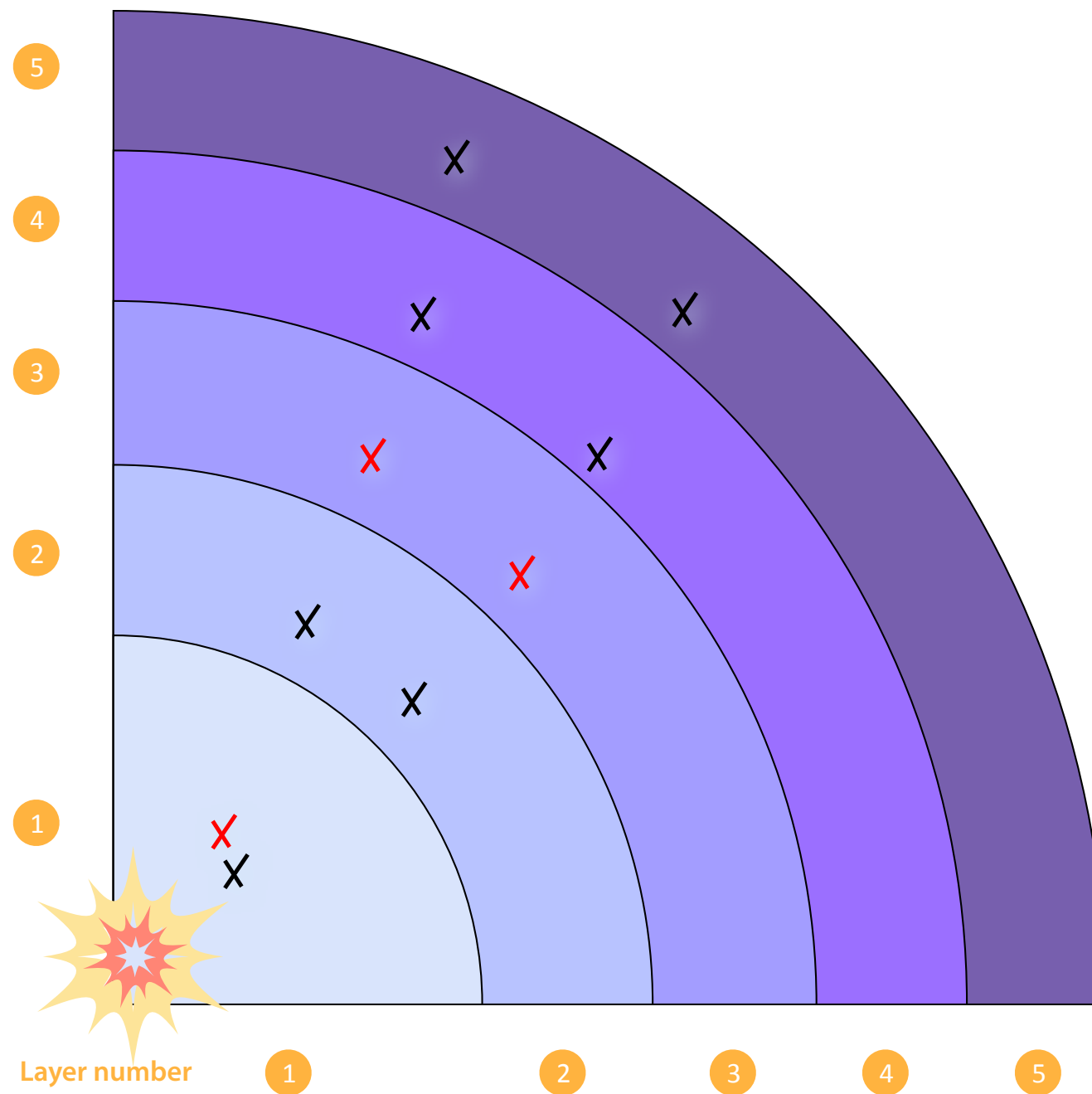
Riemann Track Finder — 1 Seeds



Riemann Track Finder — 1 Seeds

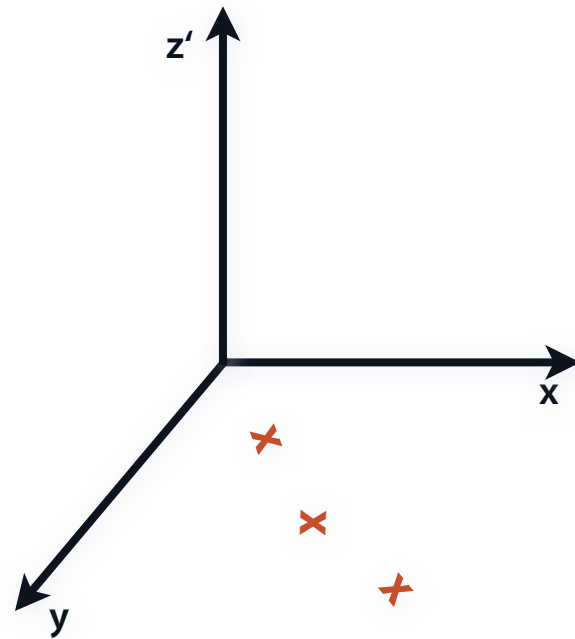


Riemann Track Finder — 1 Seeds



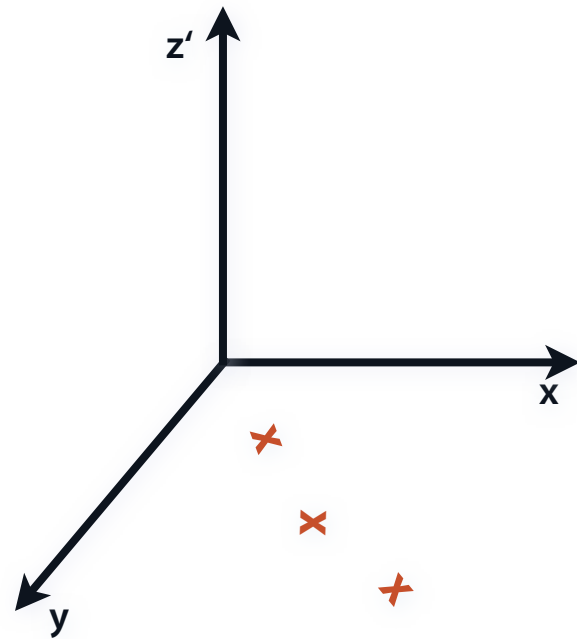
Riemann Algorithm — 2 Expansion

Riemann Algorithm — 2 Expansion

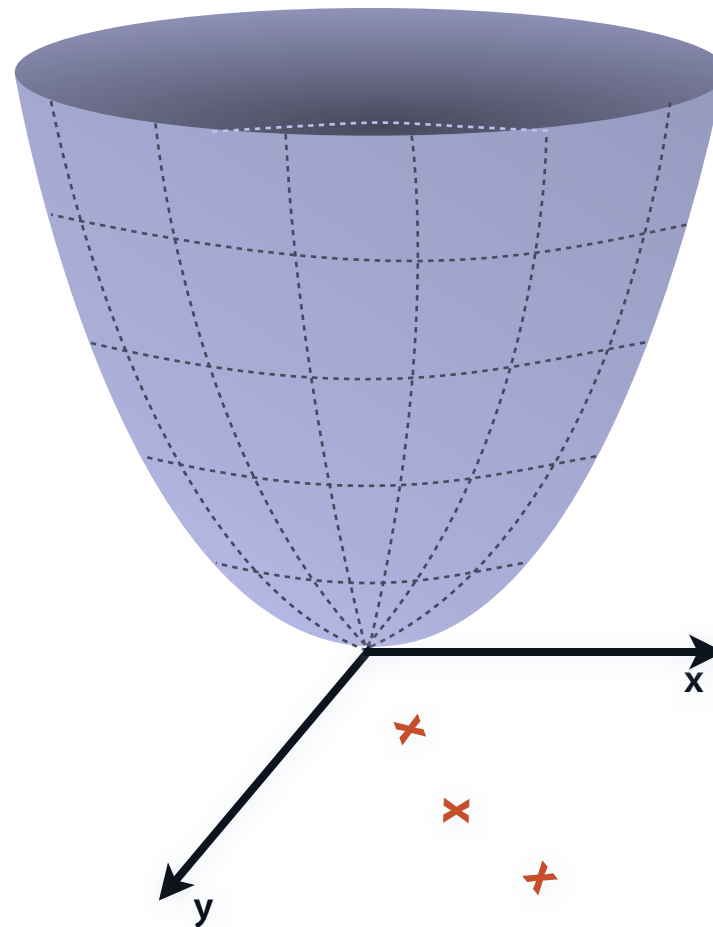


Expand to z'

Riemann Algorithm — 2 Expansion

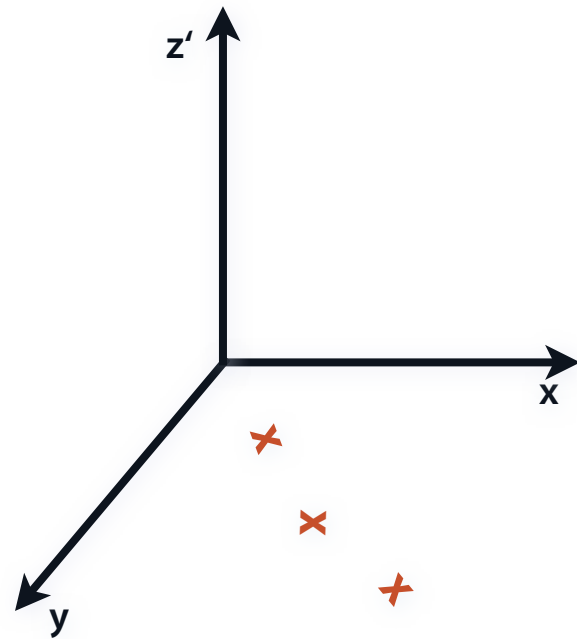


Expand to z'

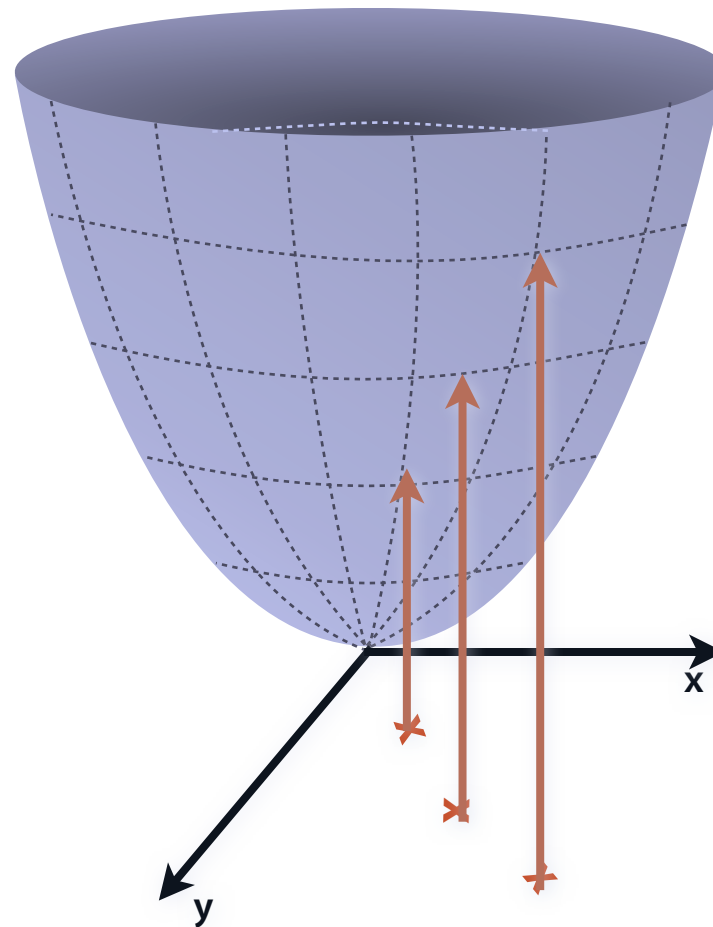


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

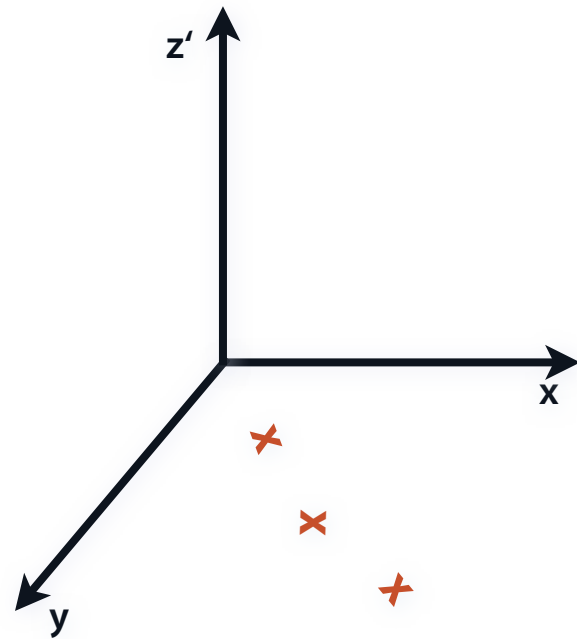


Expand to z'

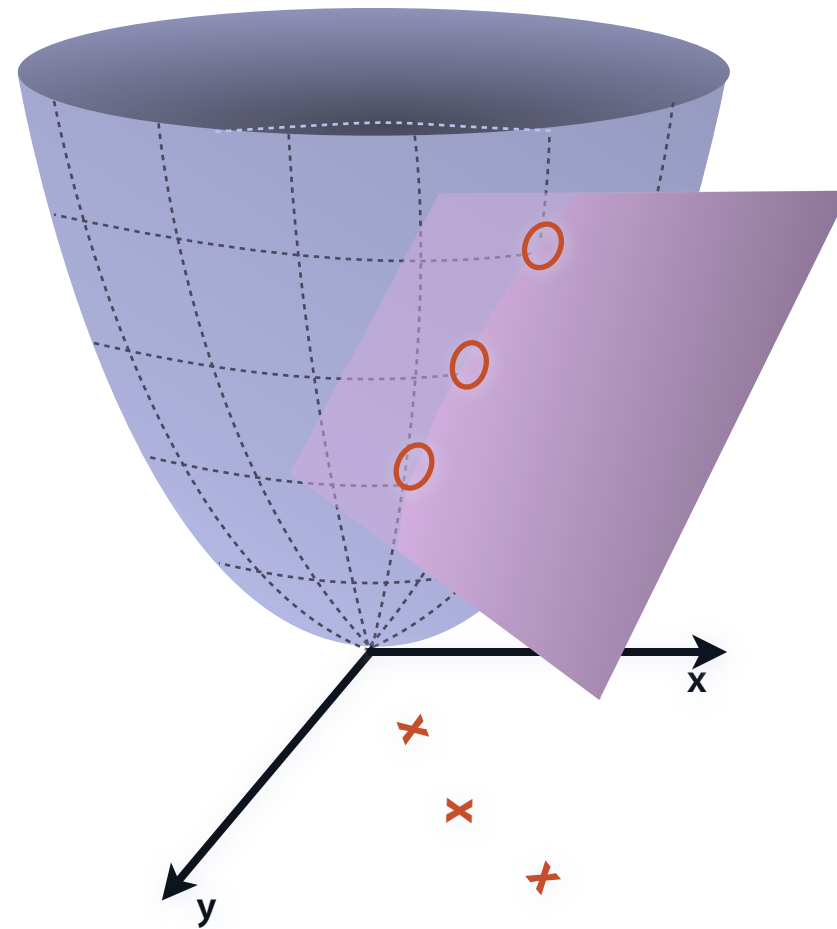


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

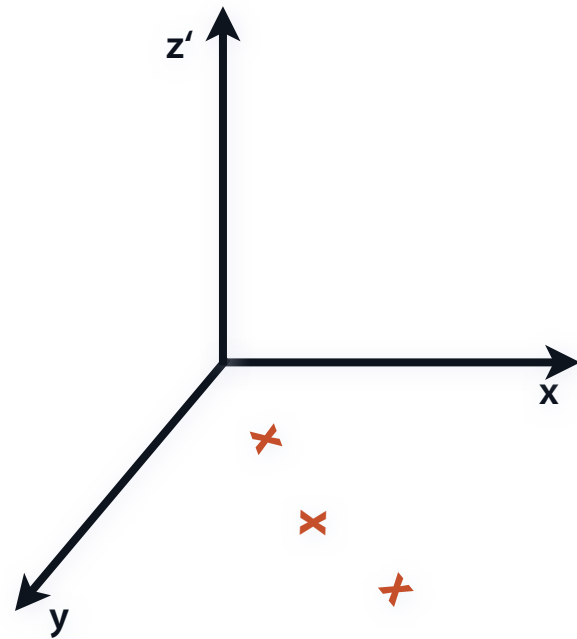


Expand to z'

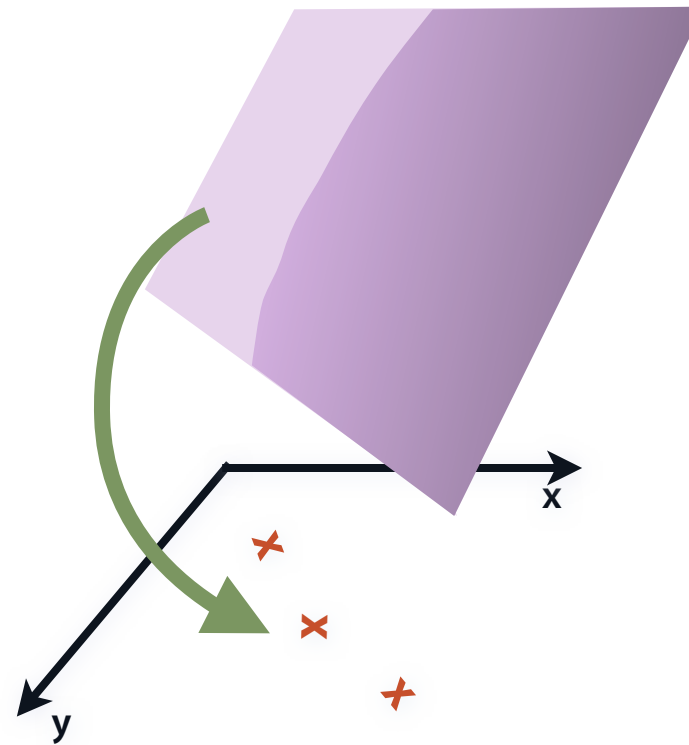


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

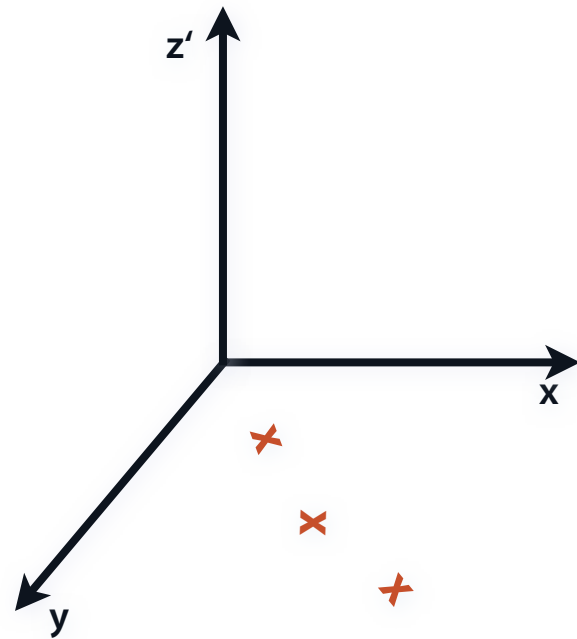


Expand to z'

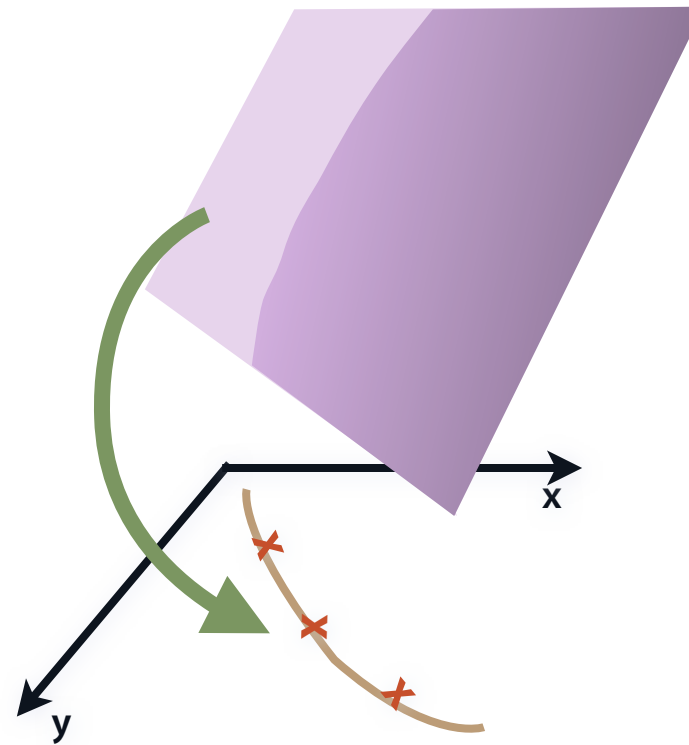


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

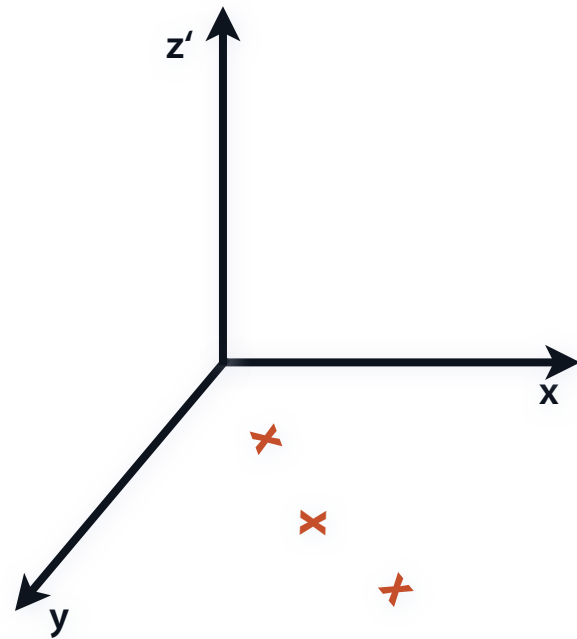


Expand to z'

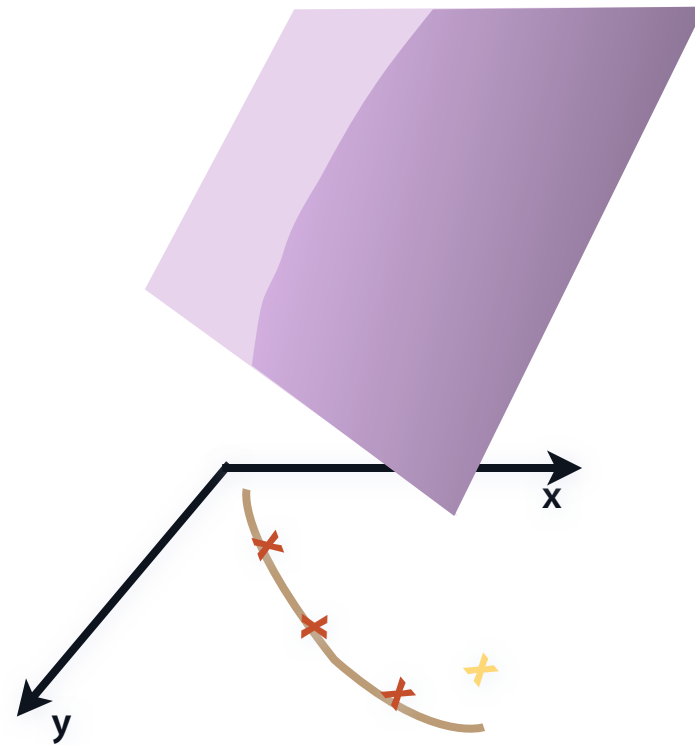


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

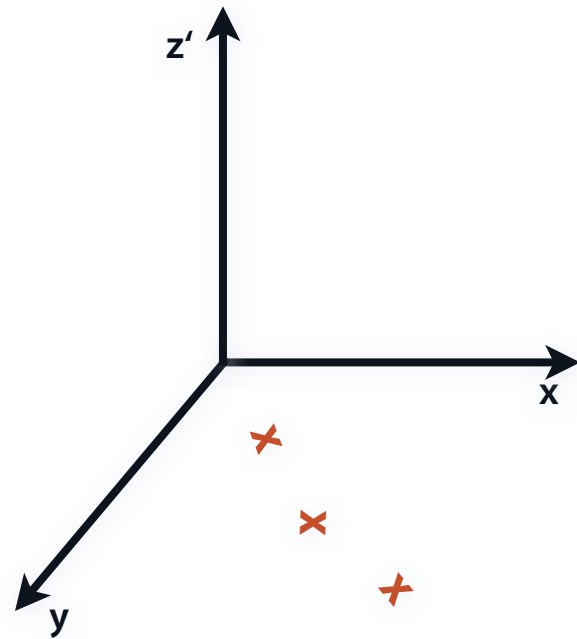


Expand to z'

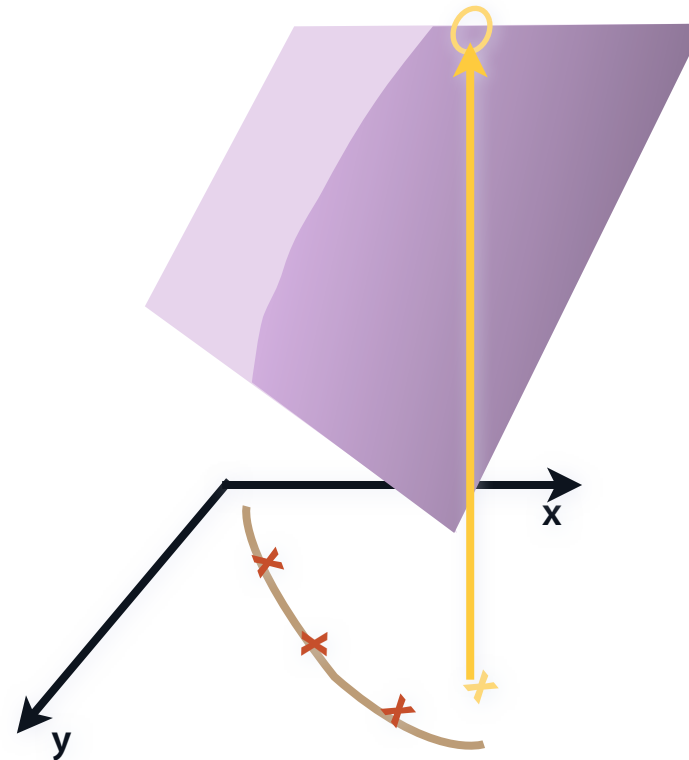


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

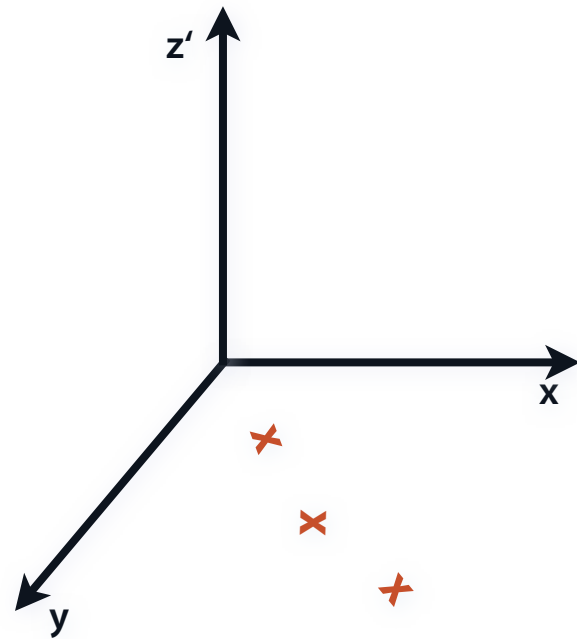


Expand to z'

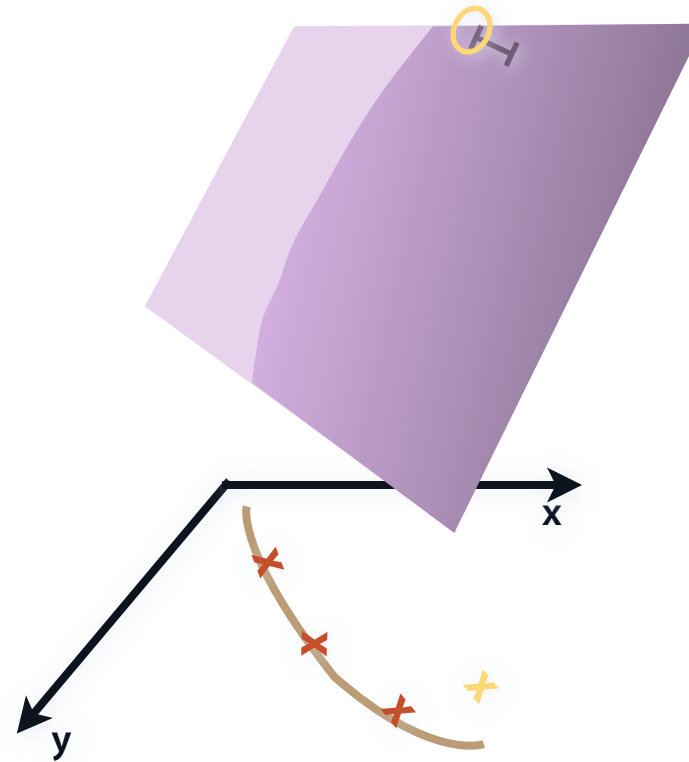


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

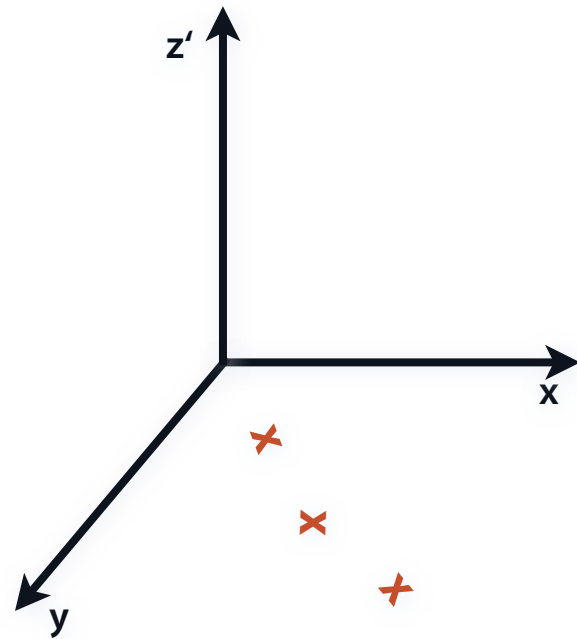


Expand to z'

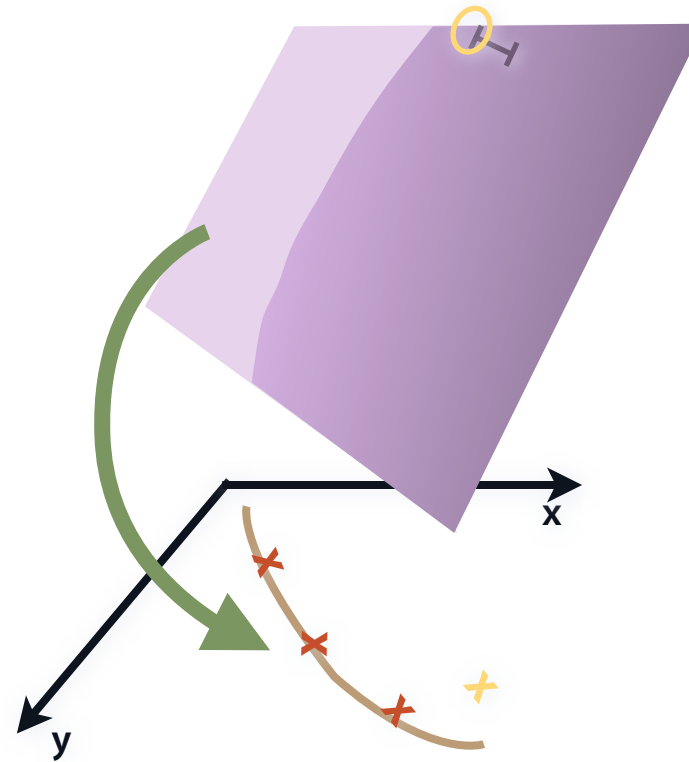


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion

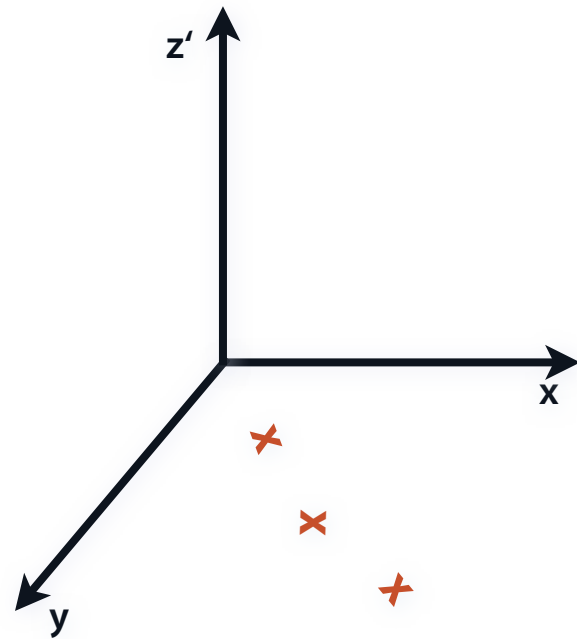


Expand to z'

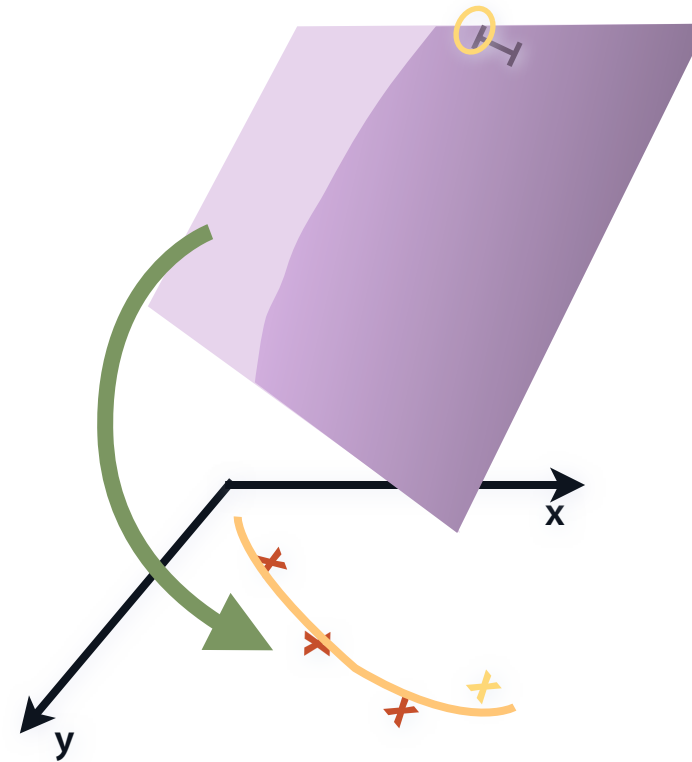


Riemann Surface
(paraboloid)

Riemann Algorithm — 2 Expansion



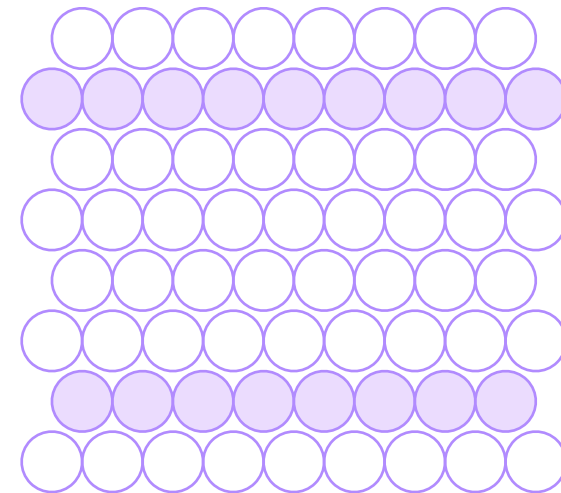
Expand to z'



Riemann Surface
(paraboloid)

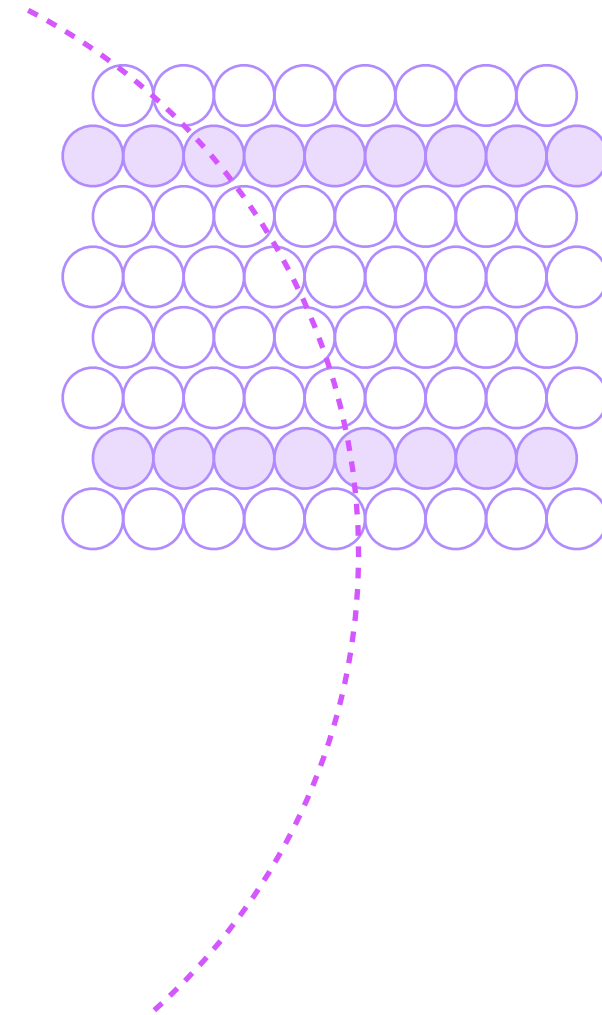
Triplet Finder — Method

STT



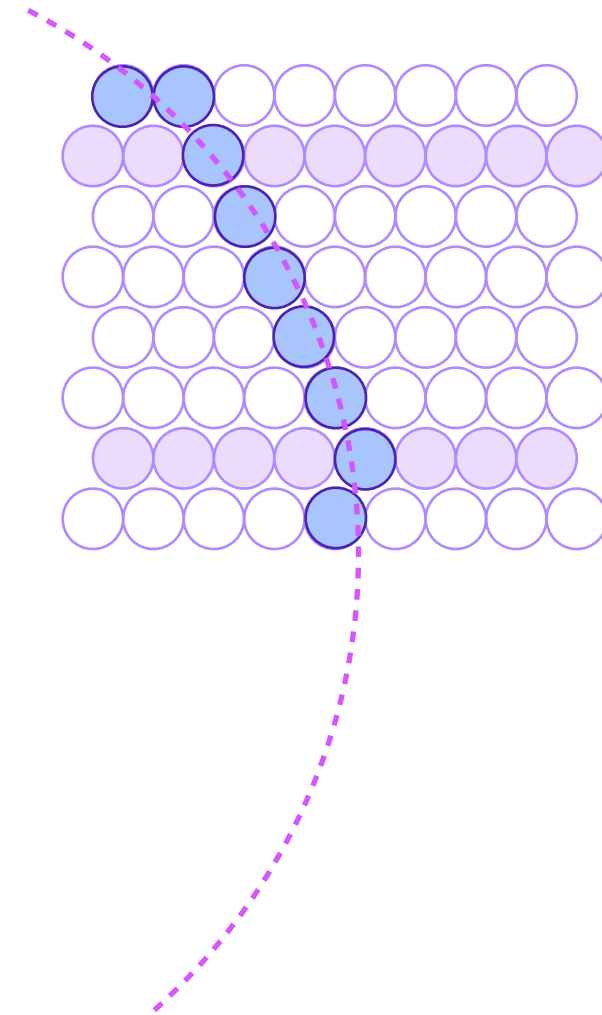
Triplet Finder — Method

STT



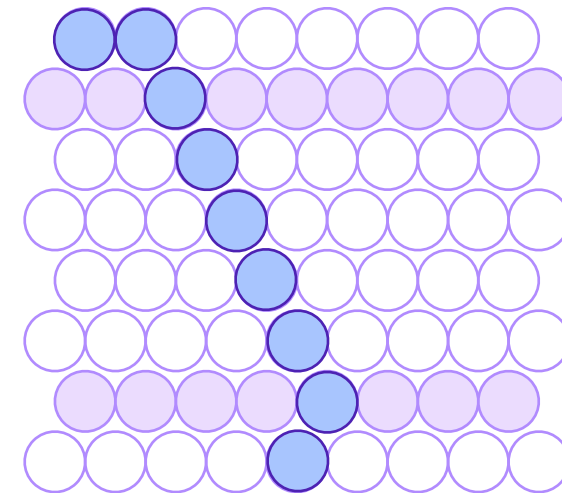
Triplet Finder — Method

STT



Triplet Finder — Method

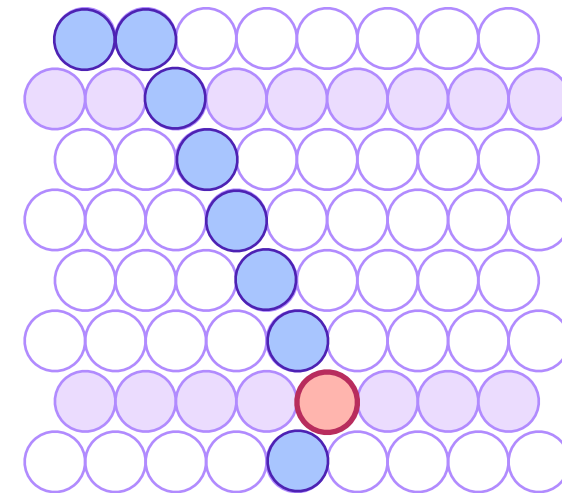
STT



Triplet Finder — Method

- STT hit in **pivot** straw

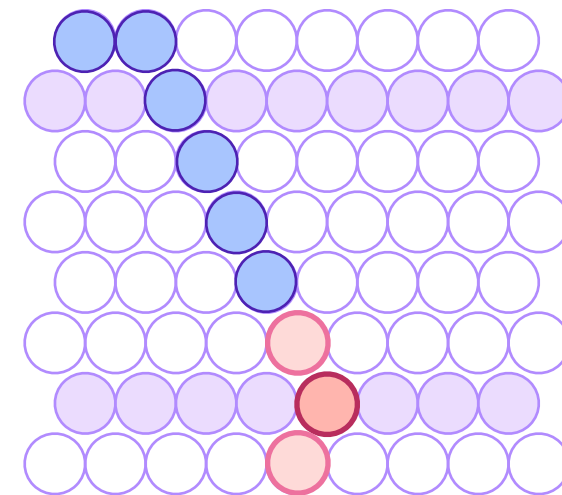
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)

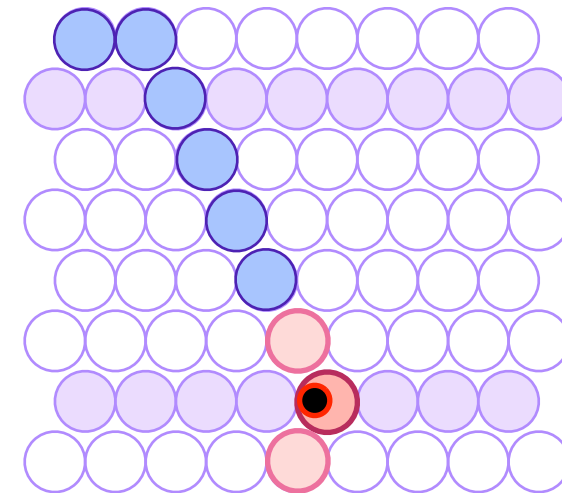
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with

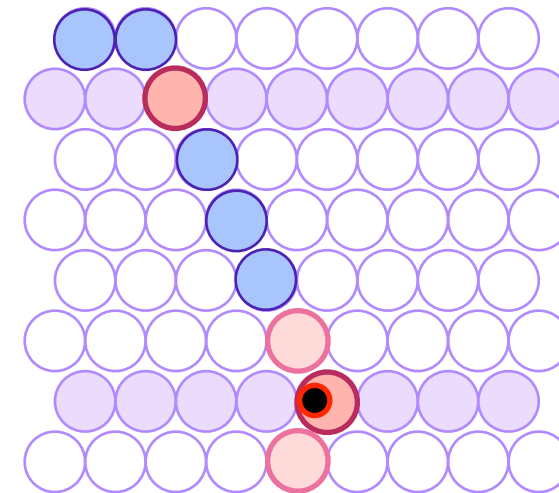
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit

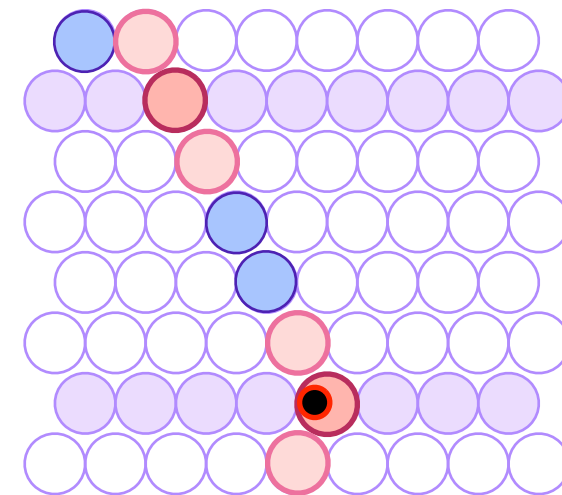
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit

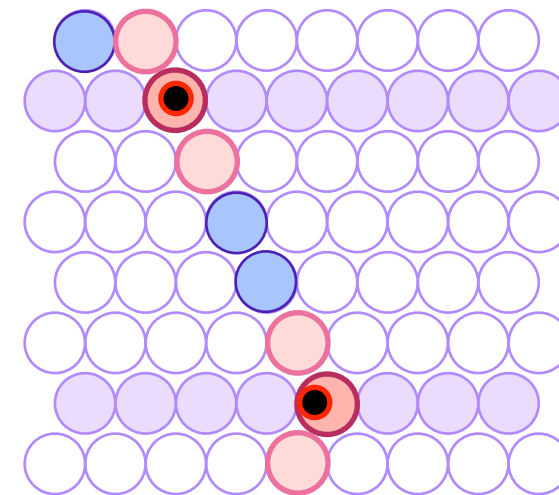
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit

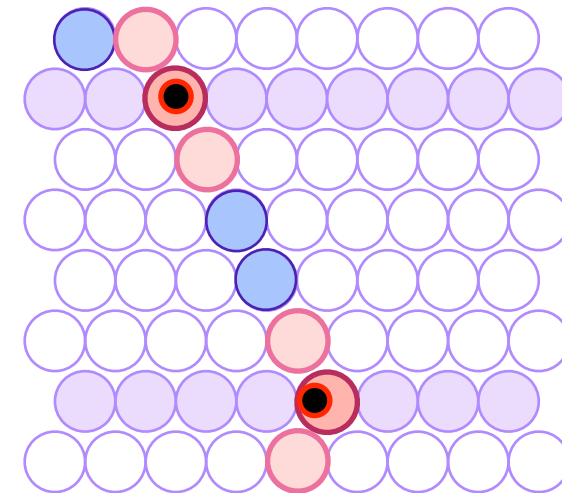
STT



Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit
 2. Interaction point

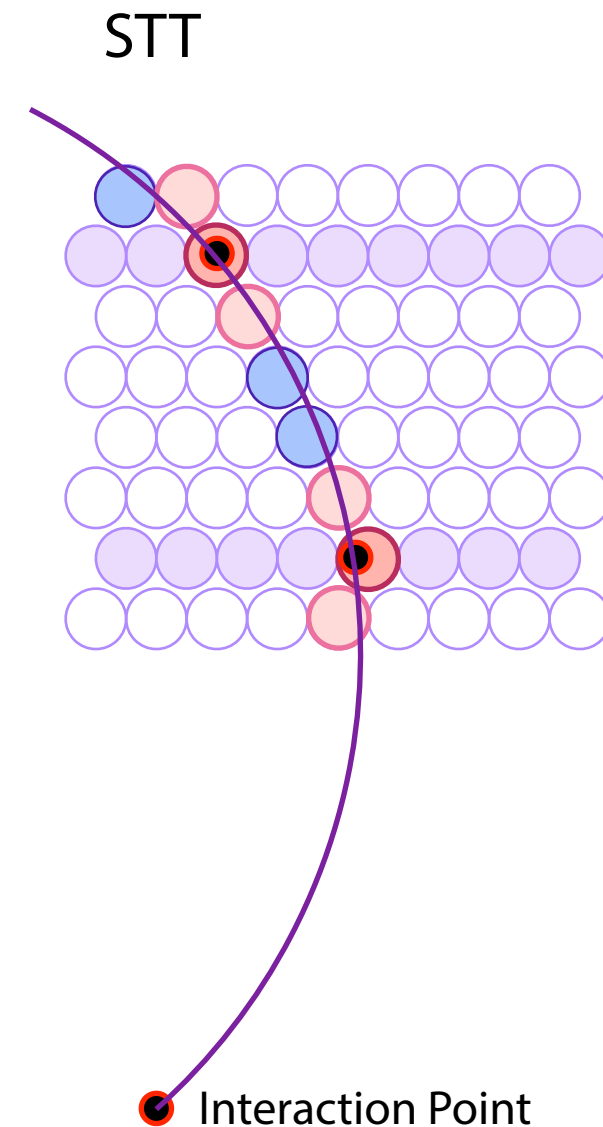
STT



● Interaction Point

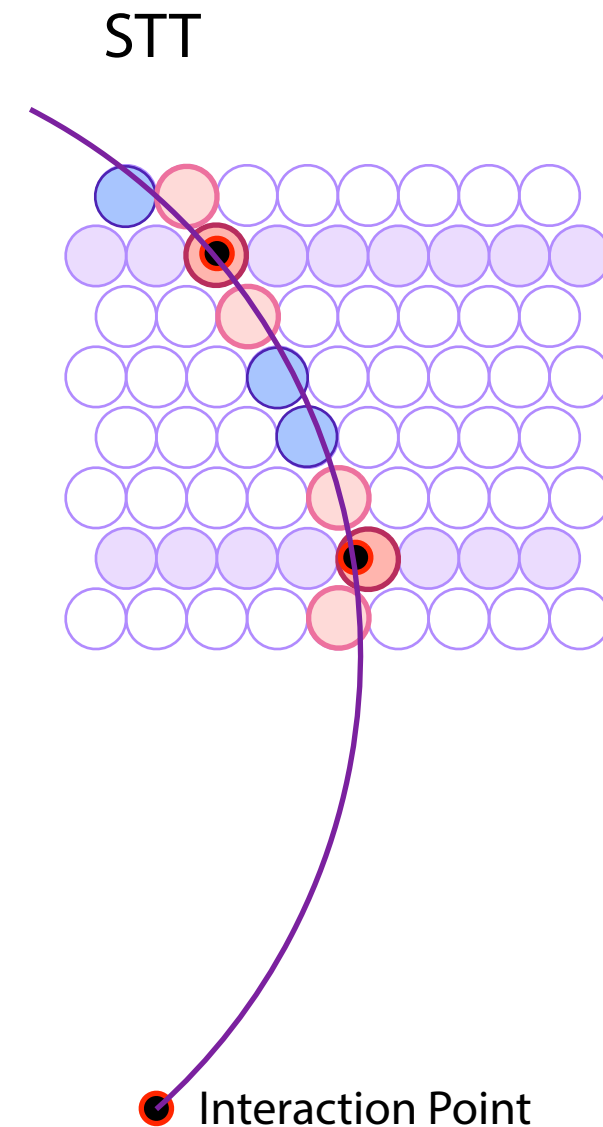
Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit
 2. Interaction point
- Calculate **circle** through three points



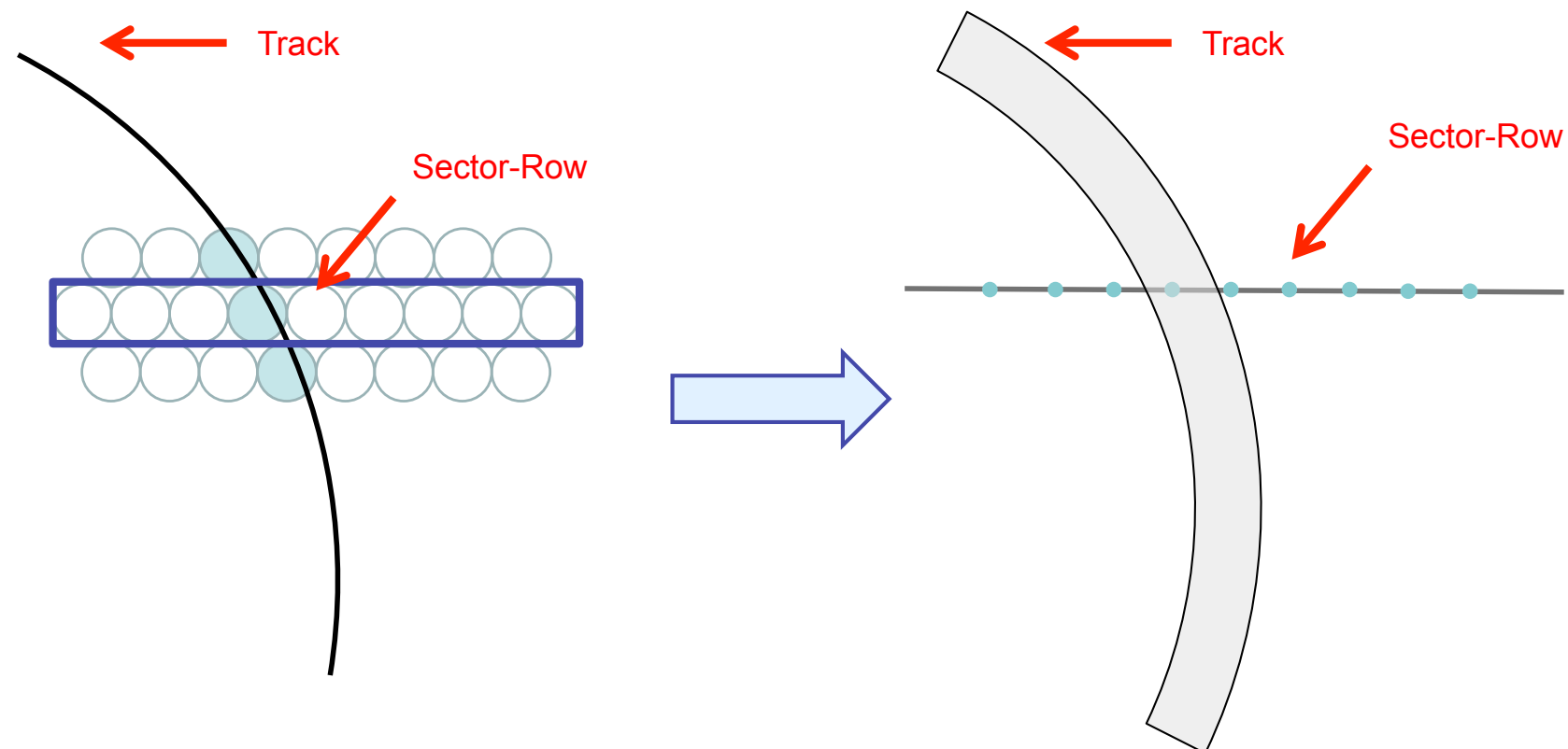
Triplet Finder — Method

- STT hit in **pivot** straw
- Find surrounding hits
 - Create **virtual hit** (*triplet*) at center of gravity (*cog*)
- **Combine** with
 1. Second STT pivot-*cog* virtual hit
 2. Interaction point
- Calculate **circle** through three points
 - **Track Candidate**



Triplet Finder — Optimizations

- Sector Row testing
 - Thicken track; shrink sector row layer to line
 - Find intersection



- Joined Kernel (*JK*): **slowest**
 - High # registers → low occupancy
- Dynamic Parallelism (*DP*) / Host Streams (*HS*): **comparable performance**
 - Performance
 - HS faster for small # processed hits, DP faster for > 45000 hits
 - HS stagnates there, while DP continues rising
 - Limiting factor
 - High # of required kernel calls
 - Kernel launch latency
 - Memcopy
 - HS more affected by this, because
 - More PCI-E transfers (launch configurations for kernels)
 - Less launch throughput, kernel launch latency gets more important
 - False dependencies of launched kernels
 - Single CPU thread handles all CUDA streams (Multi-thread possible, but synchronization overhead too high for good performance)
 - Grid scheduling done on hardware (Grid Management Unit) (DP: software)
 - » False dependencies when $N(\text{streams}) > N(\text{device connections})=32_{3.5}$

Triplet Finder — Host Stream Connections

