Event Reconstruction of
Heavy Ion Collision with TPC

고려대학교 이정우

HIM @ IBS 2016.11.26

About this talk

TPC Software

General job

- Find track with momentum and PID.
- Simulation.

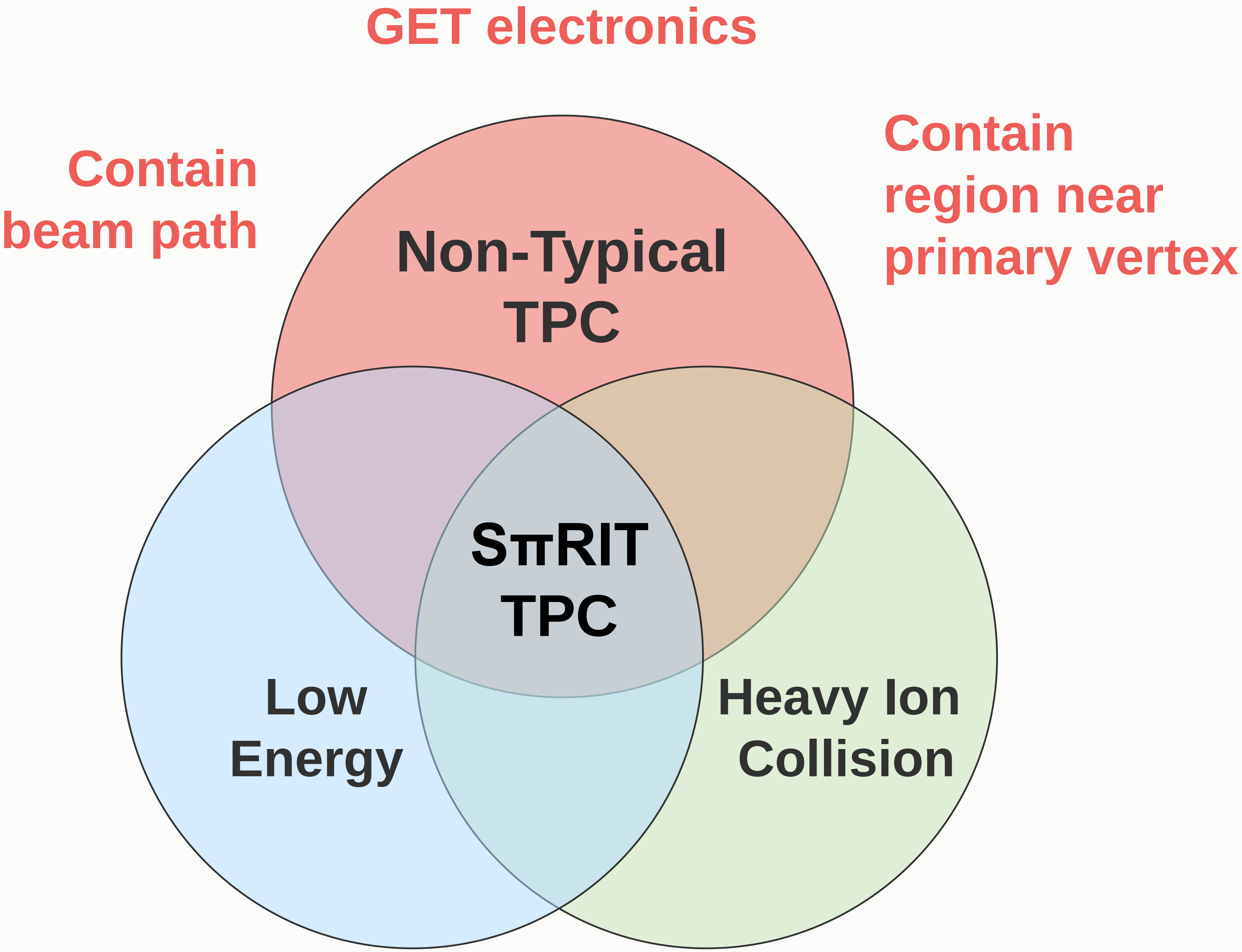
User's need

- Event display.
- Bug fix.
- Pretty pictures.
- Online analysis / event viewer.
- Documentation

Things to be managed

- Software that works on different kind of environment.
- Reconstruction speed.
- Memory management
- Size of output.

SπRIT-TPC



GET electronics

Contain
beam path

Contain
region near
primary vertex

Non-Typical
TPC

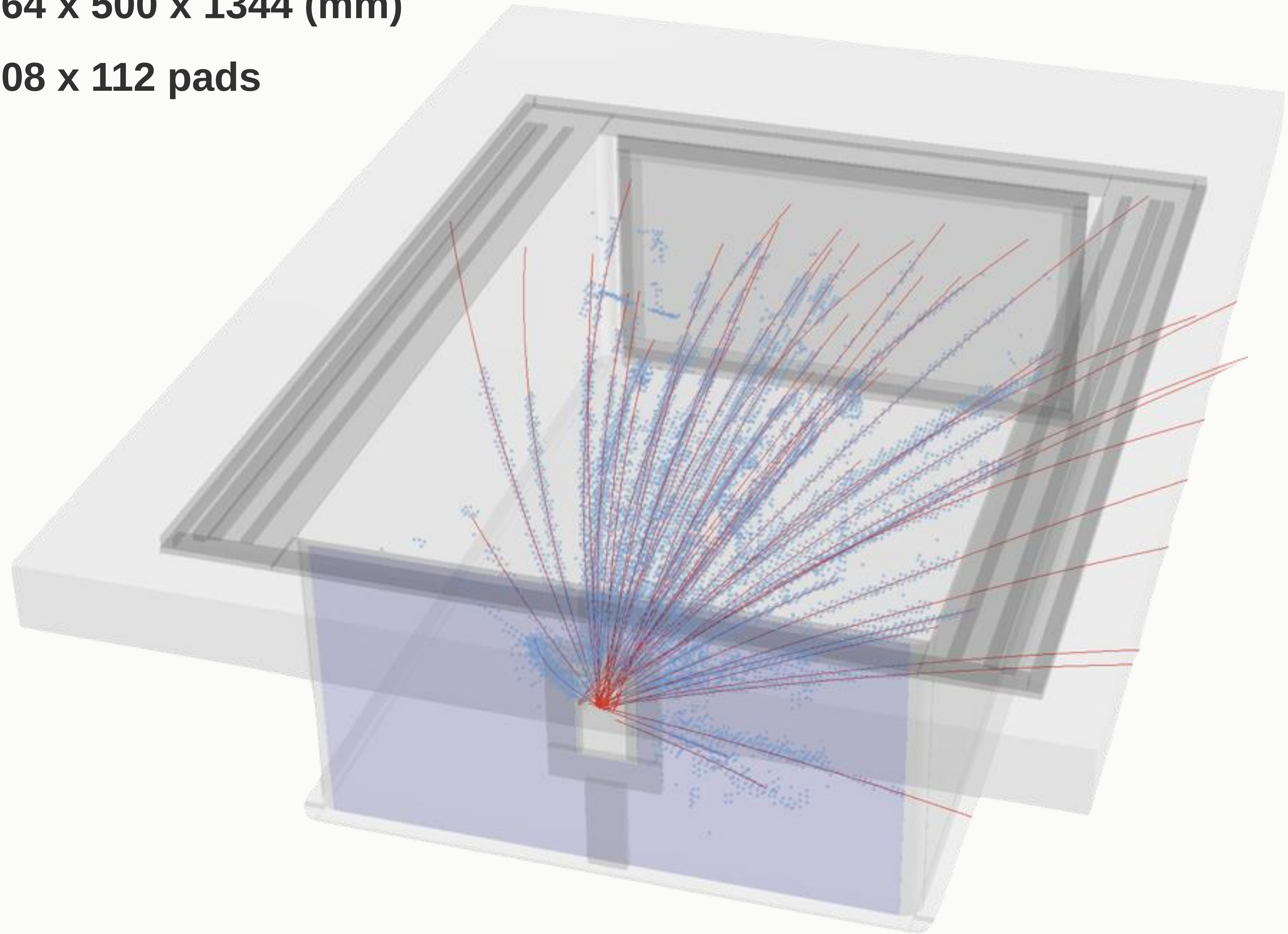
SπRIT
TPC

Low
Energy

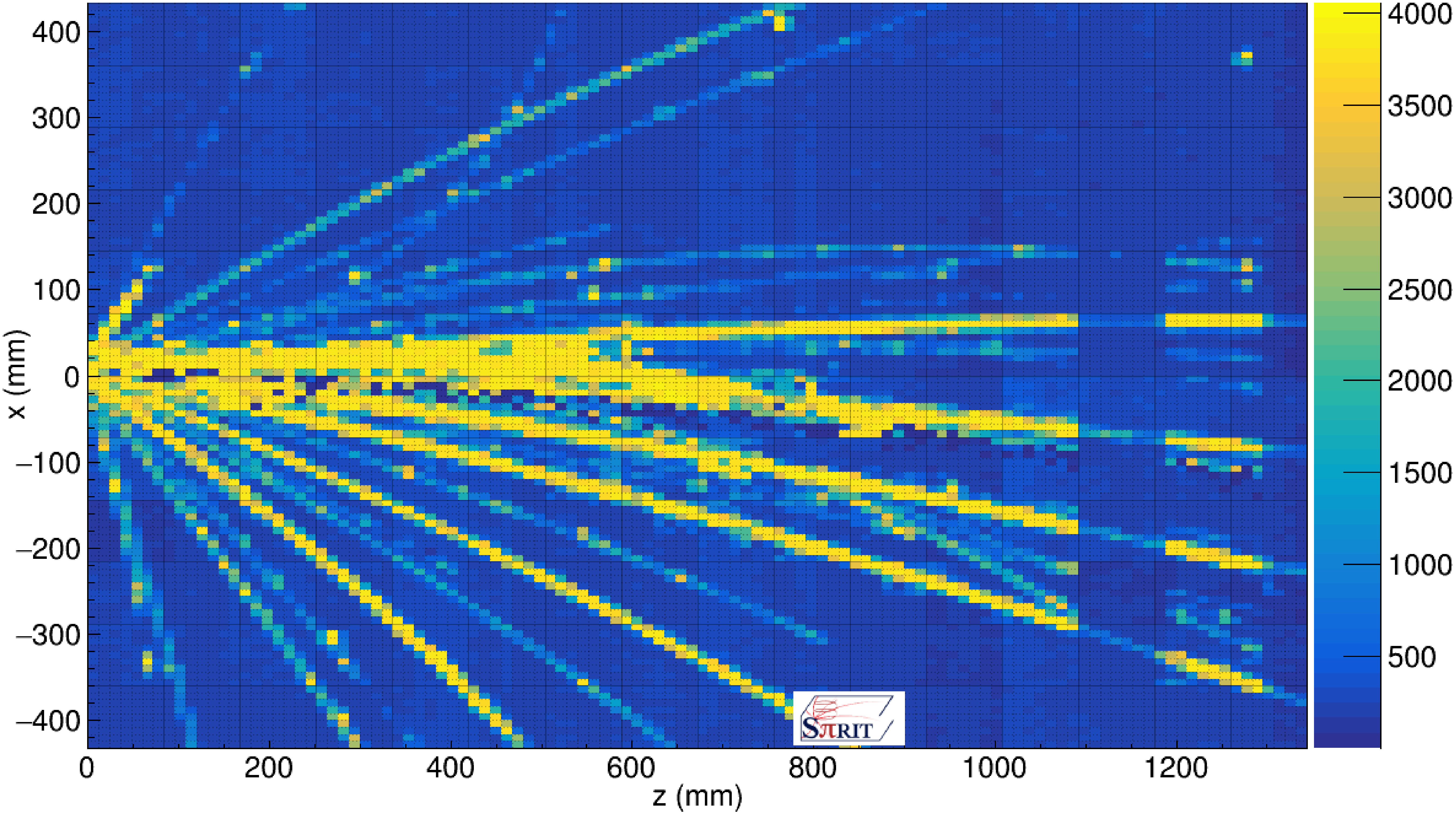
Heavy Ion
Collision

SπRIT-TPC

- **864 x 500 x 1344 (mm)**
- **108 x 112 pads**

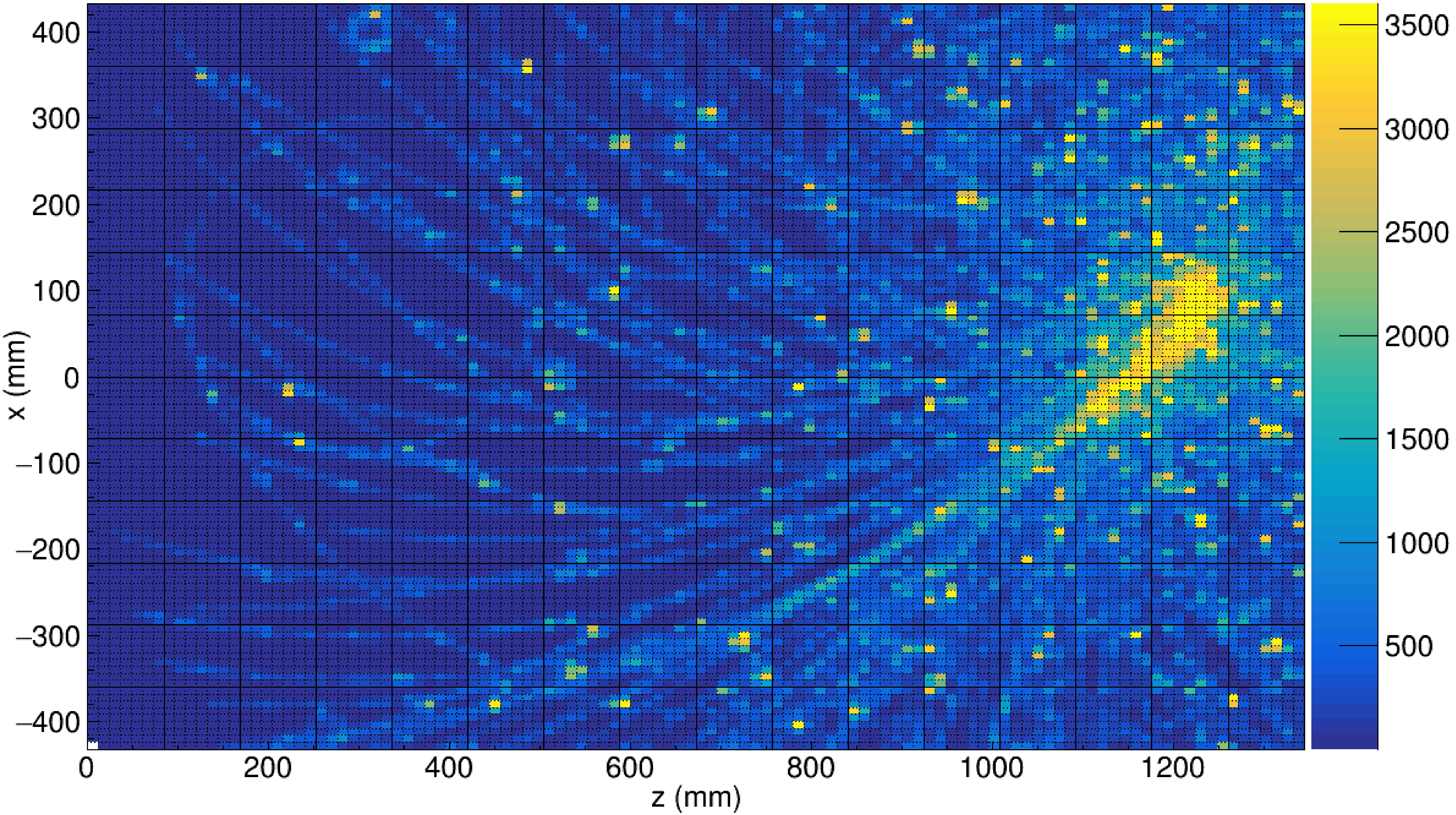


SπRIT-TPC

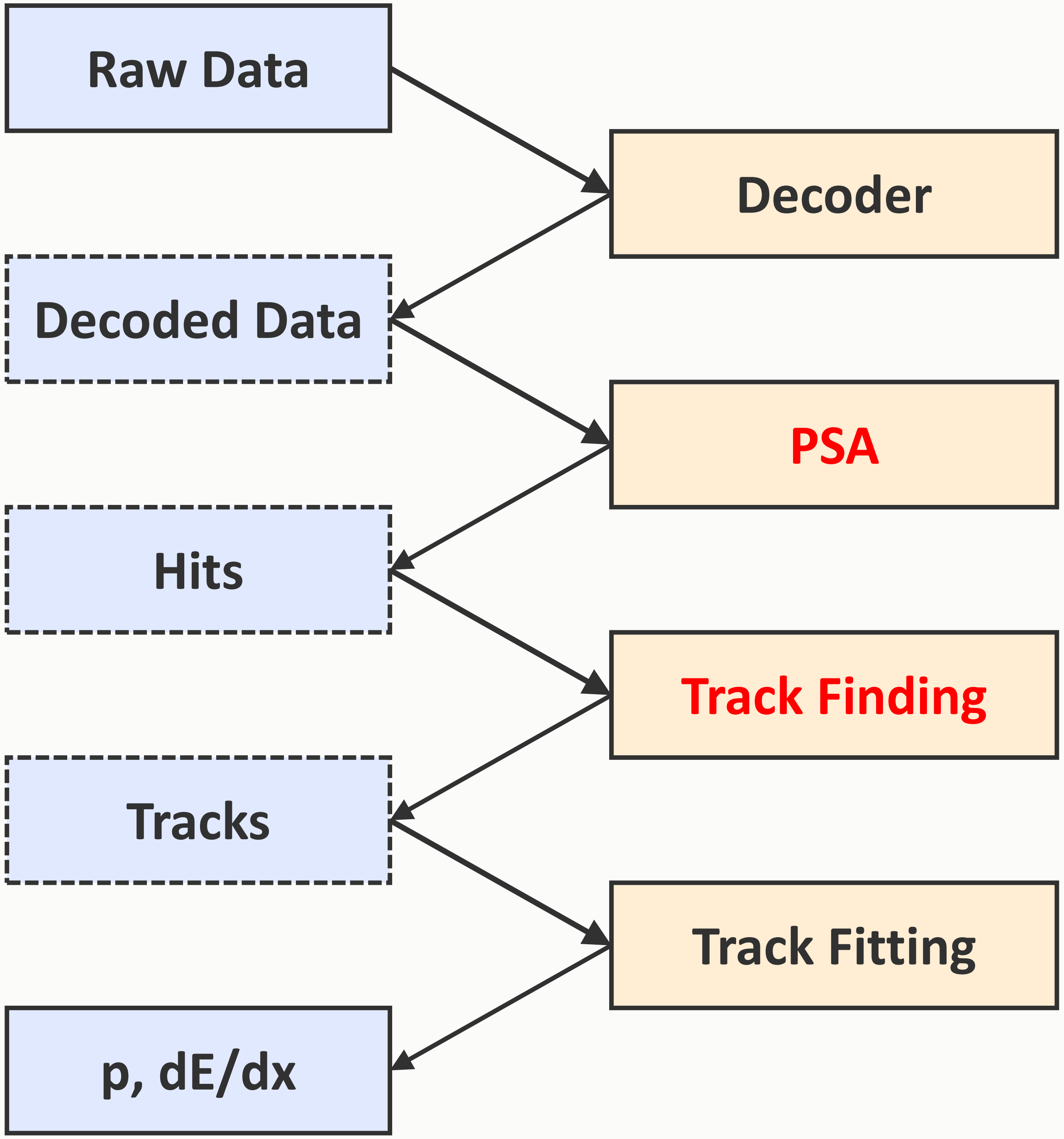


x (mm)
400
300
200
100
0
−100
−200
−300
−400
z (mm)
0
200
400
600
800
1000
1200
4000
3500
3000
2500
2000
1500
1000
500
SπRIT

SπRIT-TPC


400
300
200
100
0
−100
−200
−300
−400
x (mm)
0
200
400
600
800
1000
1200
z (mm)
3500
3000
2500
2000
1500
1000
500

TPC Software Reconstruction Flow



Importance Pulse Shape Analysis

PSA Task

- Count number of hits
- Find Charge
- Find Position

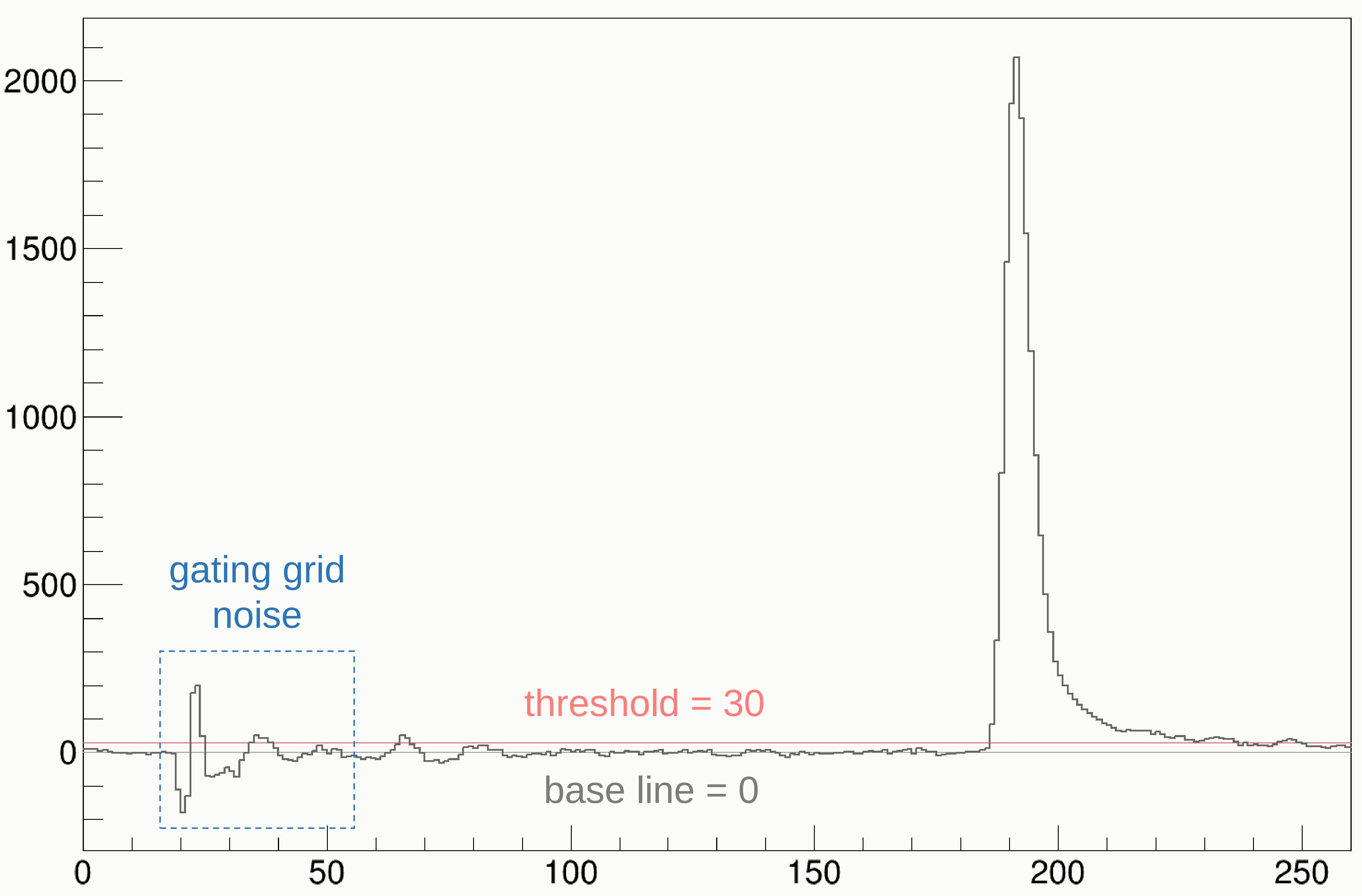
Effect of PSA

- Position resolution → Track finding efficiency → Momentum efficiency
- Charge resolution → dE/dx resolution → PID resolution

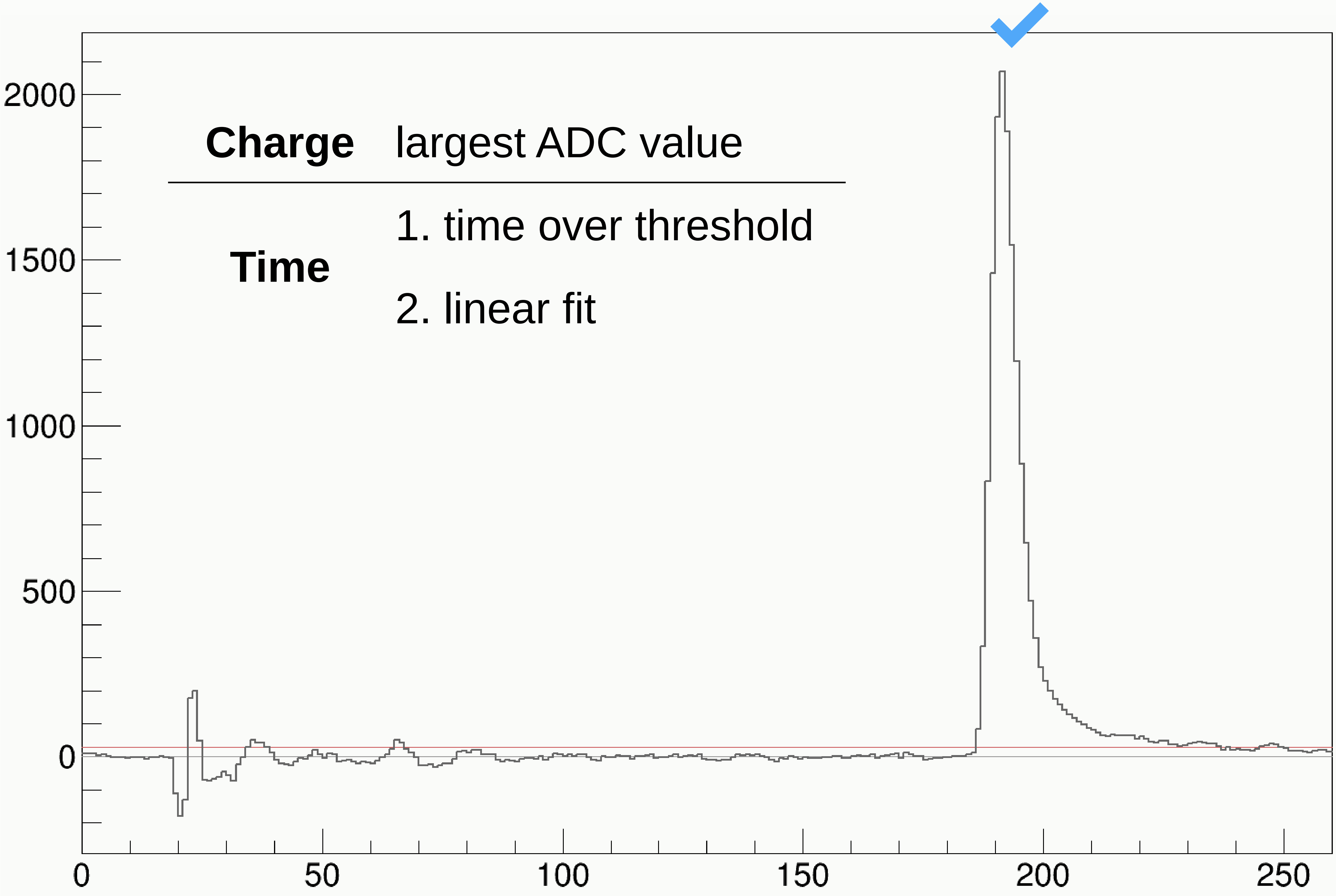
Issues

- Multi-Hit
- Saturation
- Pile-Up

Finding Single-Hit

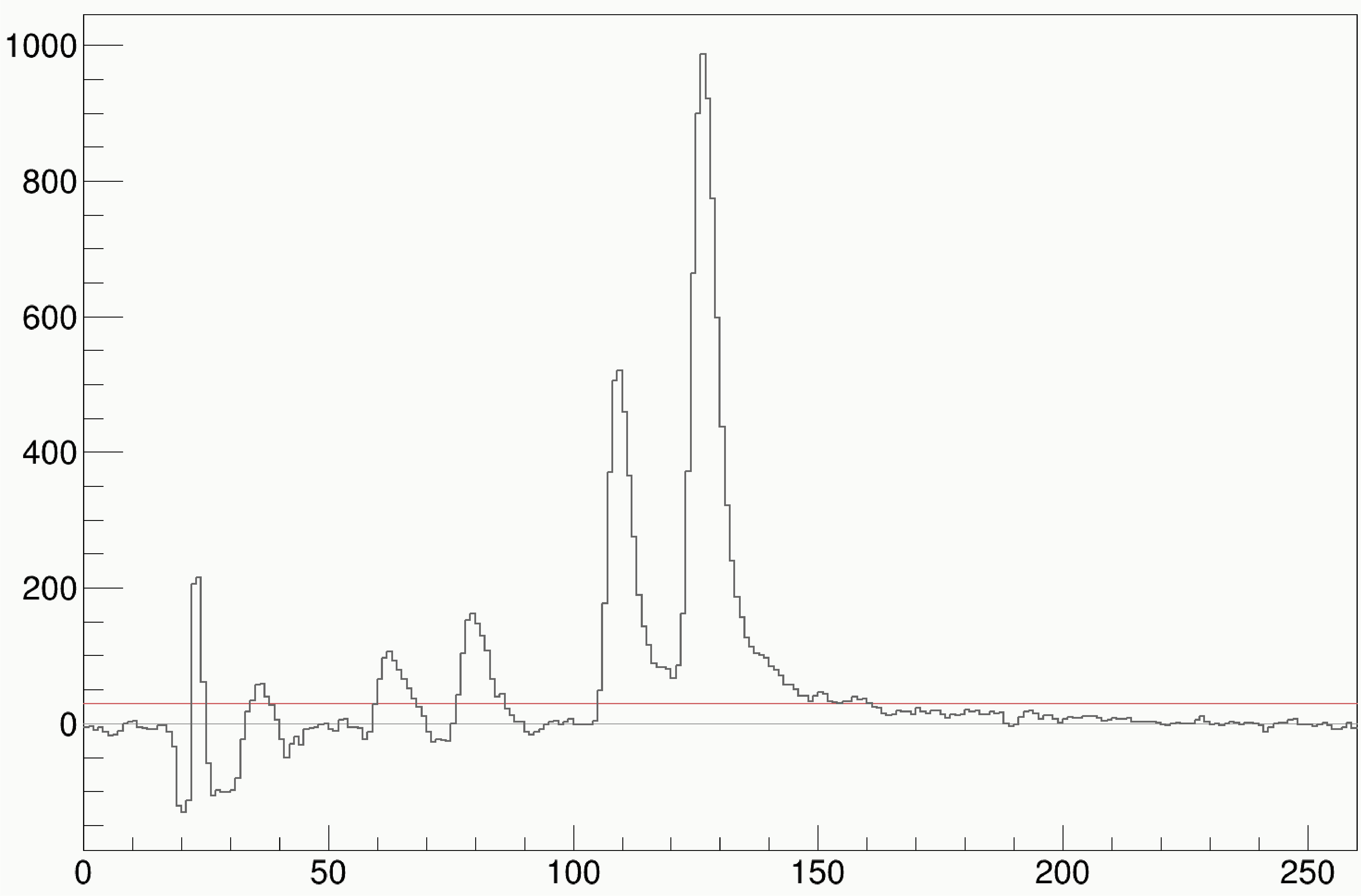


Finding Single-Hit

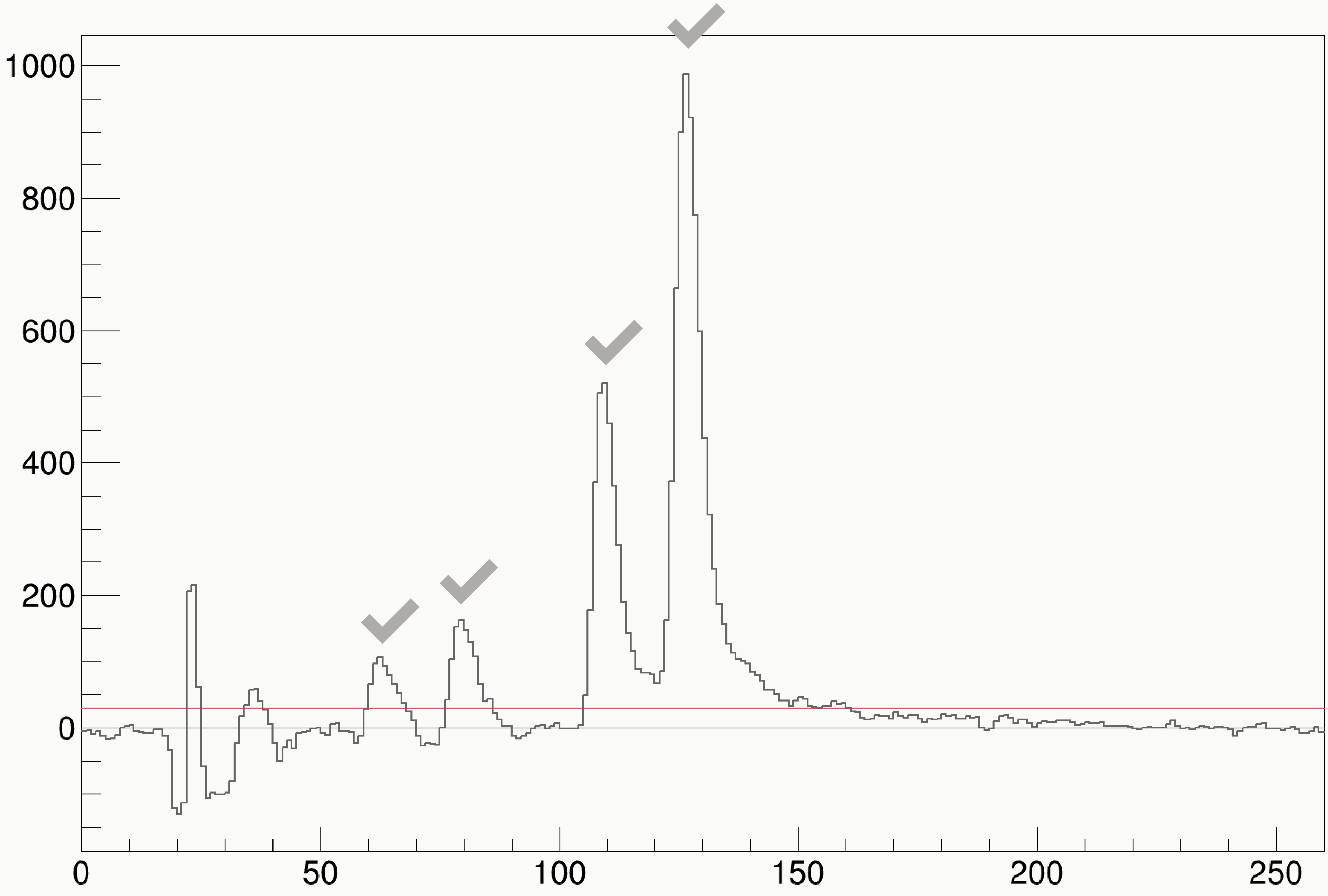


Charge	largest ADC value
Time	1. time over threshold 2. linear fit

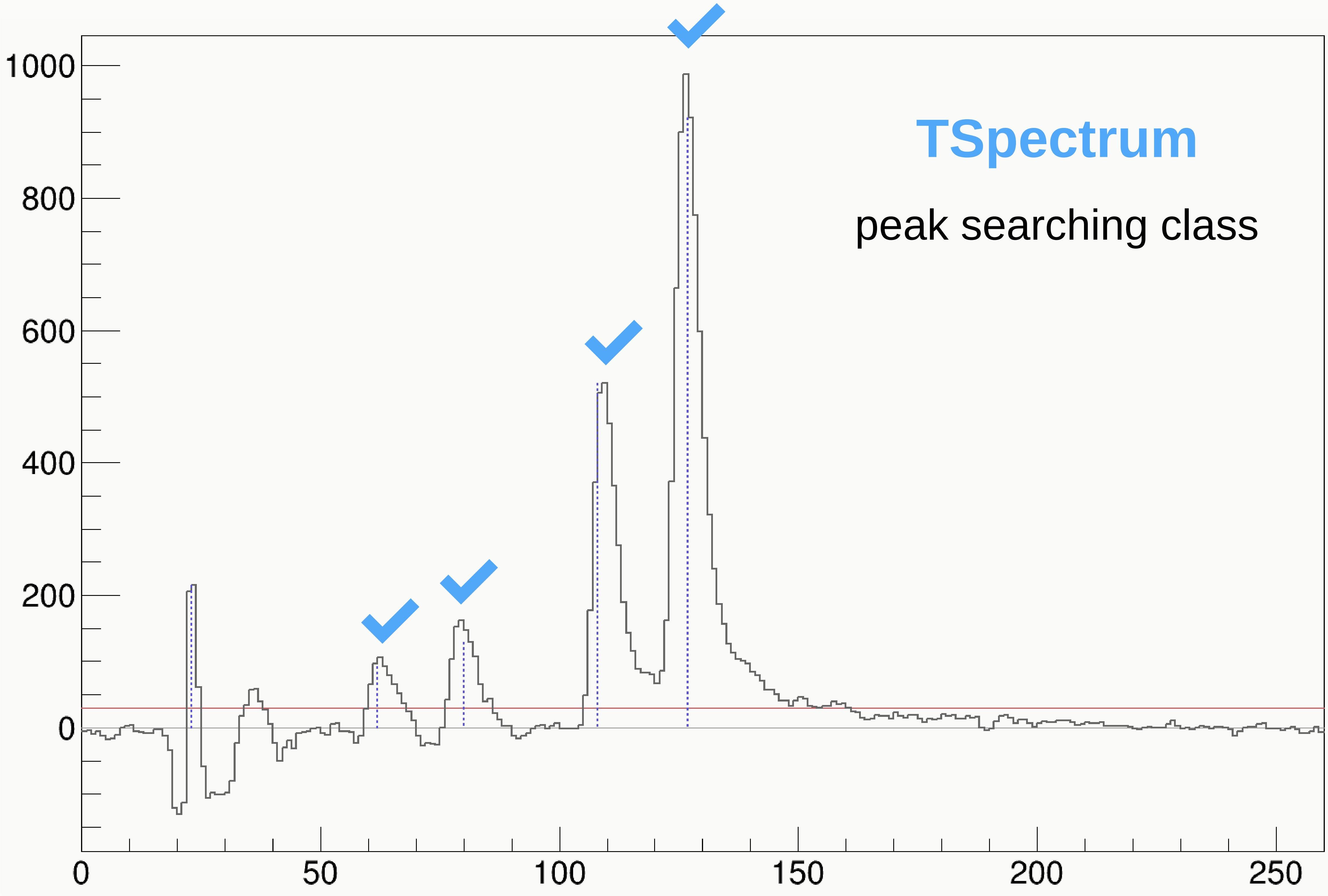
Finding Multi-Hits



Finding Multi-Hits



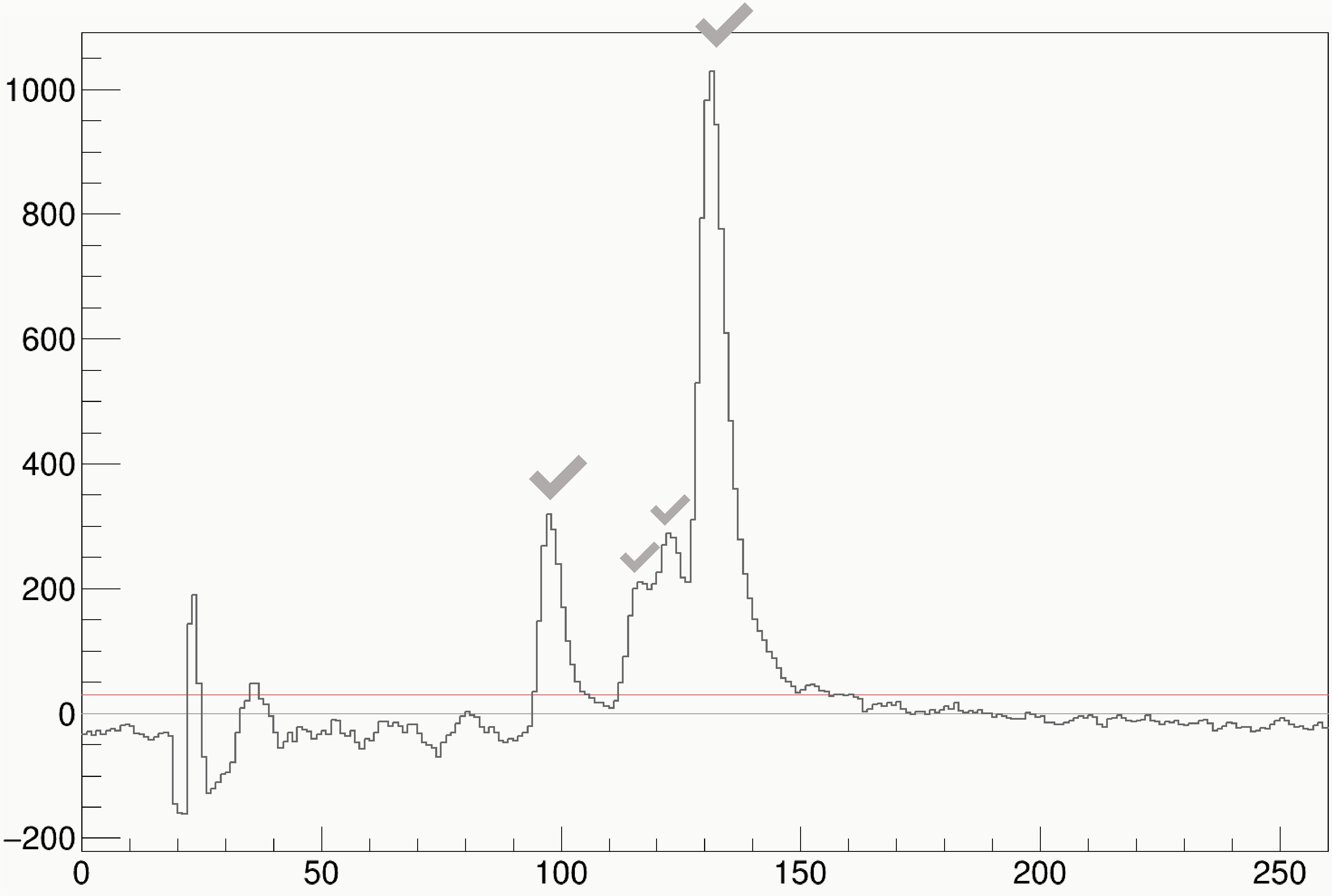
Finding Multi-Hits



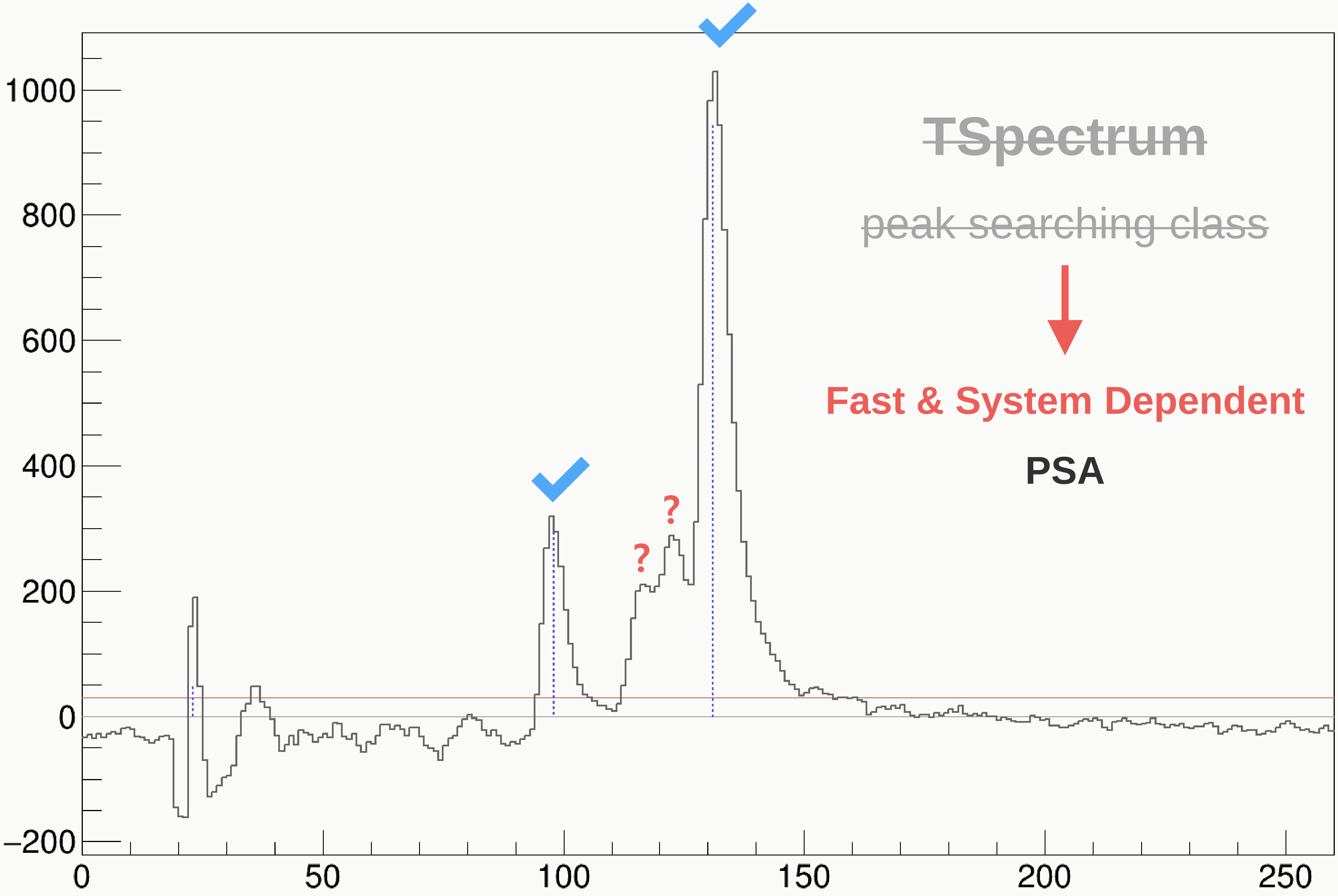
TSpectrum

peak searching class

Finding Multi-Hits



Finding Multi-Hits



~~TSpectrum~~

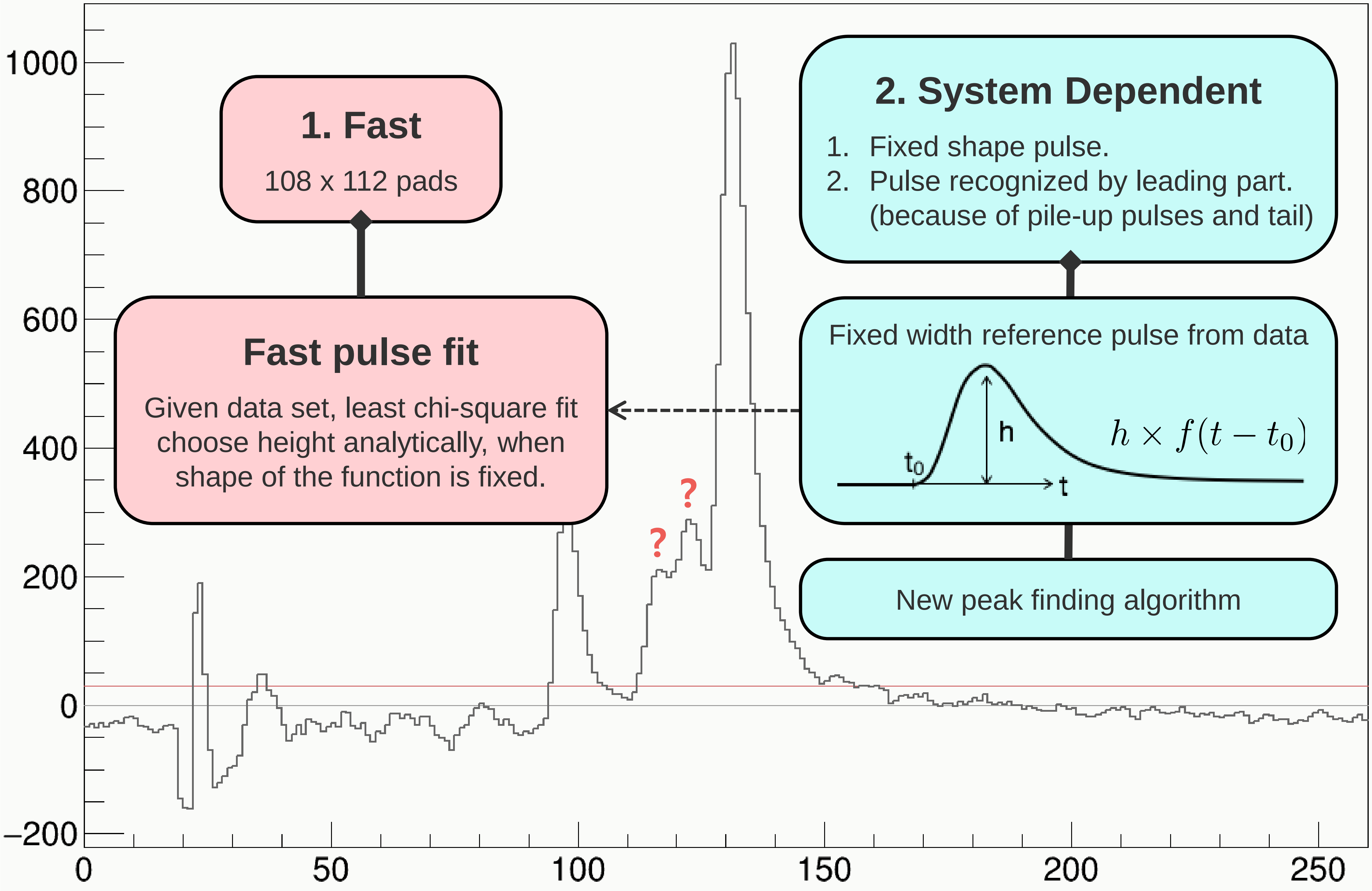
~~peak searching class~~

↓

Fast & System Dependent

PSA

Fast & System Dependent PSA



1. Fast

108 x 112 pads

Fast pulse fit

Given data set, least chi-square fit choose height analytically, when shape of the function is fixed.

2. System Dependent

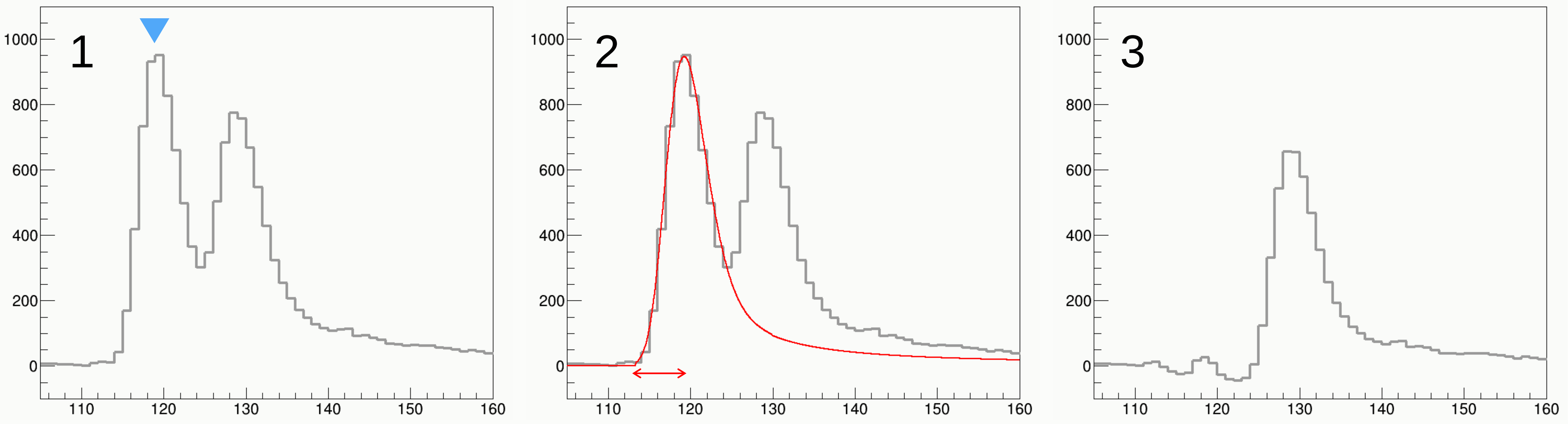
1. Fixed shape pulse.
2. Pulse recognized by leading part. (because of pile-up pulses and tail)

Fixed width reference pulse from data

$h \times f(t - t_0)$

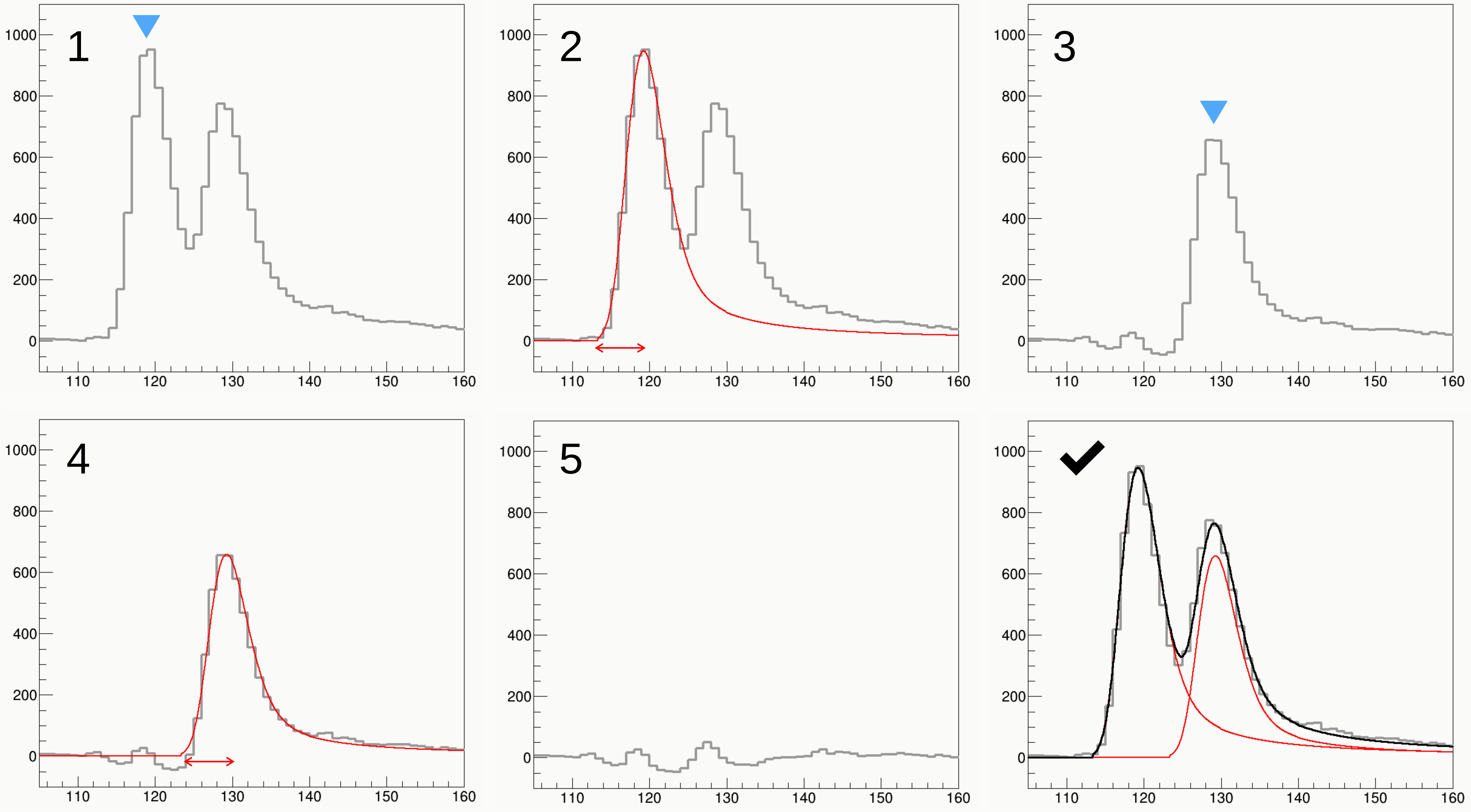
New peak finding algorithm

Peak Finding Algorithm

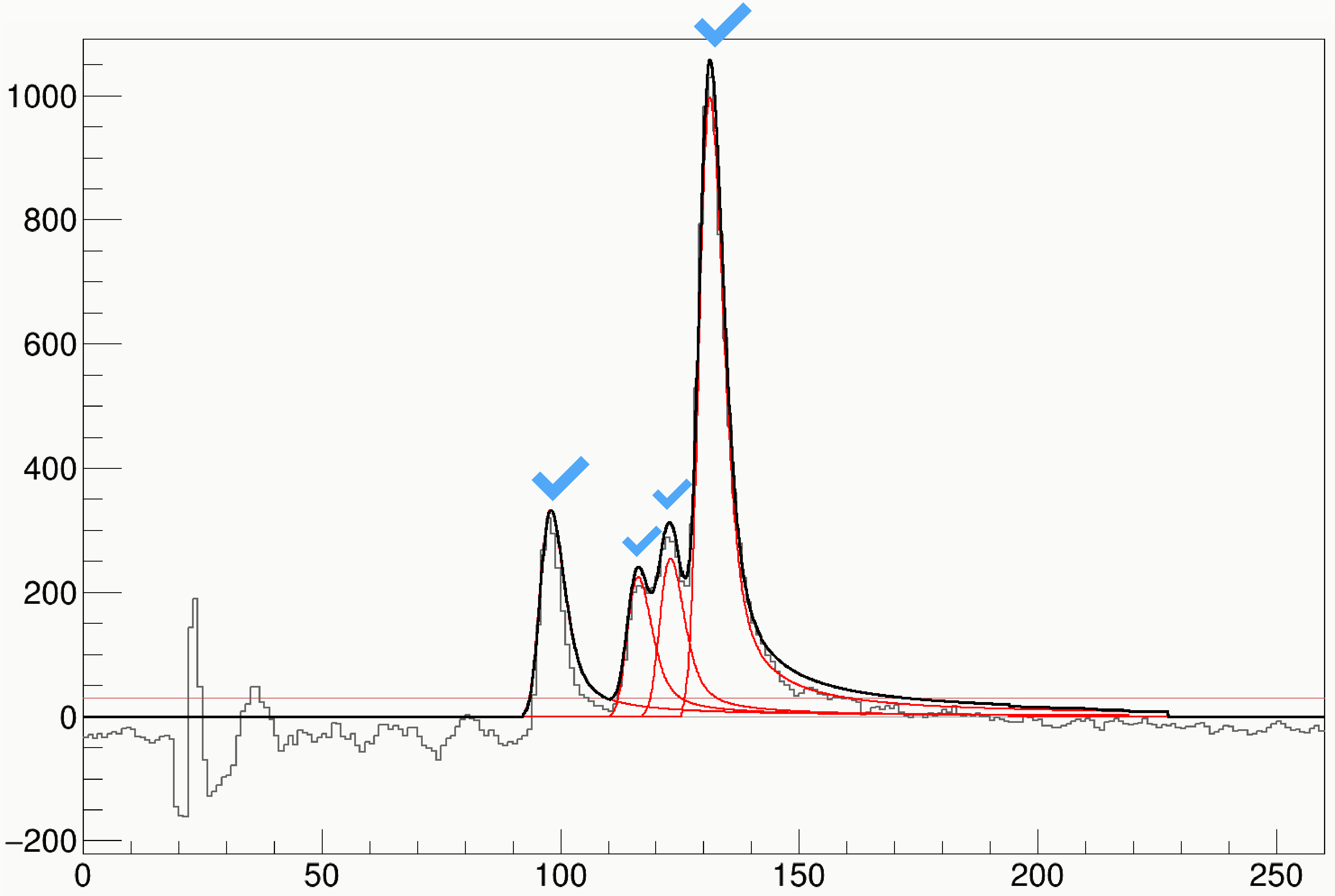


- While scanning time-bucket increasing order :
  1. If value of ADC(>threshold) increase 4 times in a row.
  2. Fit rising part of pulse with reference pulse.
  3. Subtract found pulse from original distribution.
- Do 1. ~ 3. until the end of channel.

Peak Finding Algorithm



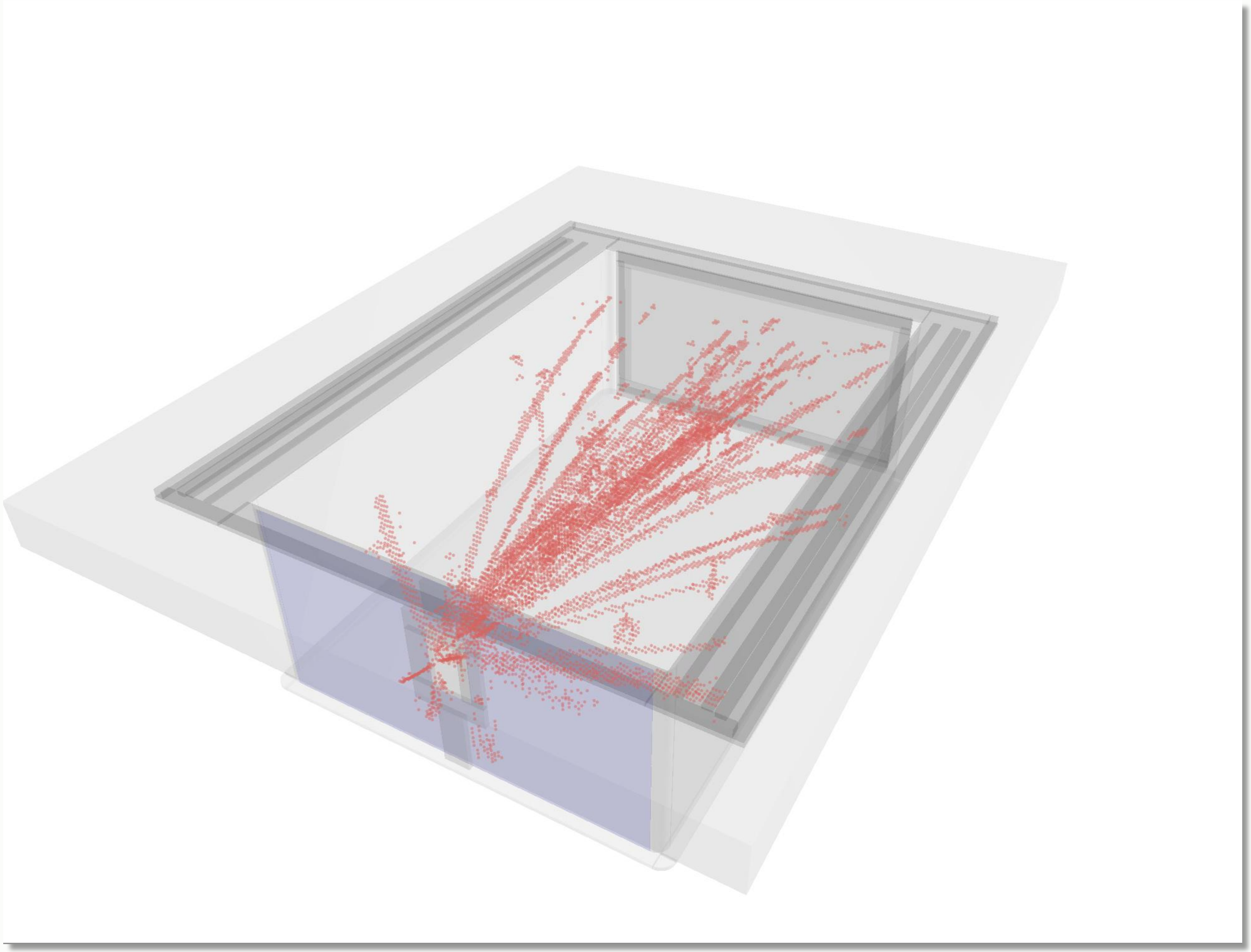
Current PSA



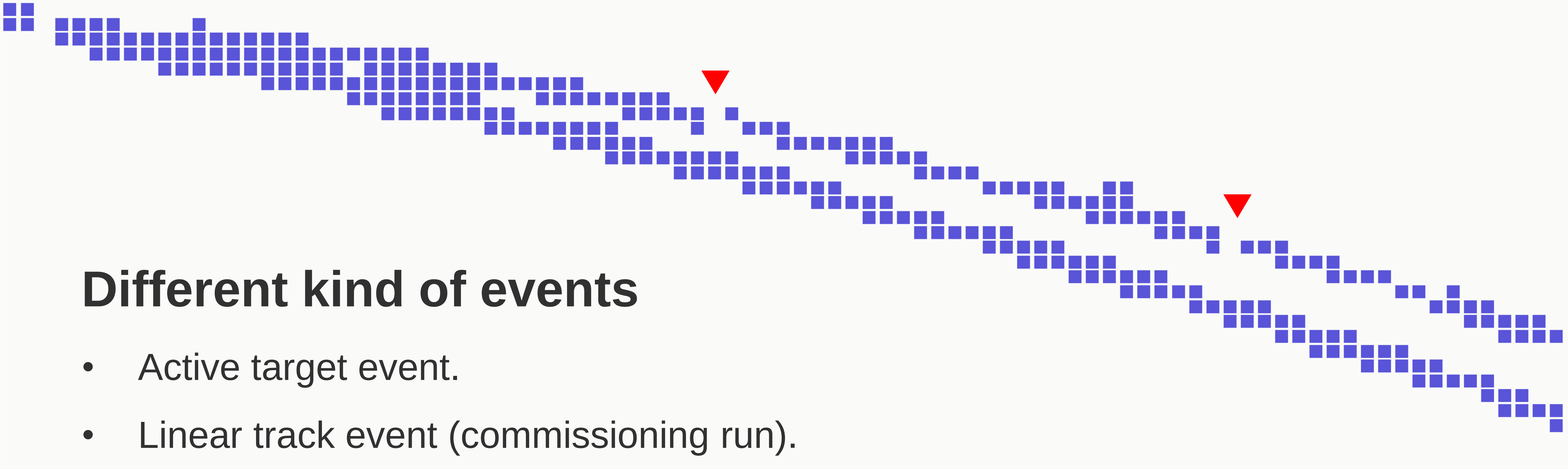
Summary for PSA

- GET electronics was provided with no solution to reconstruct hit.
- We developed multi-hit finding PSA task with fixed width pulse.
- But recent results show pulse shape is not fixed. Shape depend on track angle, electron drift length and amount of input charge.
- More analysis needs to be done on pad saturation.

Result after PSA



Tracking Issues



Different kind of events

- Active target event.
- Linear track event (commissioning run).
- Cosmic event (with or without B-field).

Different kind of tracks

- Thickness.
- Dip-angle (angle between p and p_T).
- Beam.

Empty points

- Saturation by heavy ions.
- Dead channels along beam path.
- Delta electrons.
- Track with large dip angle.
- Etc.

Time Line

Riemann Tracking

: Direct implementation from FOPI software.

Cosmic Test (2015 Summer)

Linear Tracking

: Tracking for linear track events (cosmic, commissioning)

Commissioning Run (October, 2015)

Curve Tracking

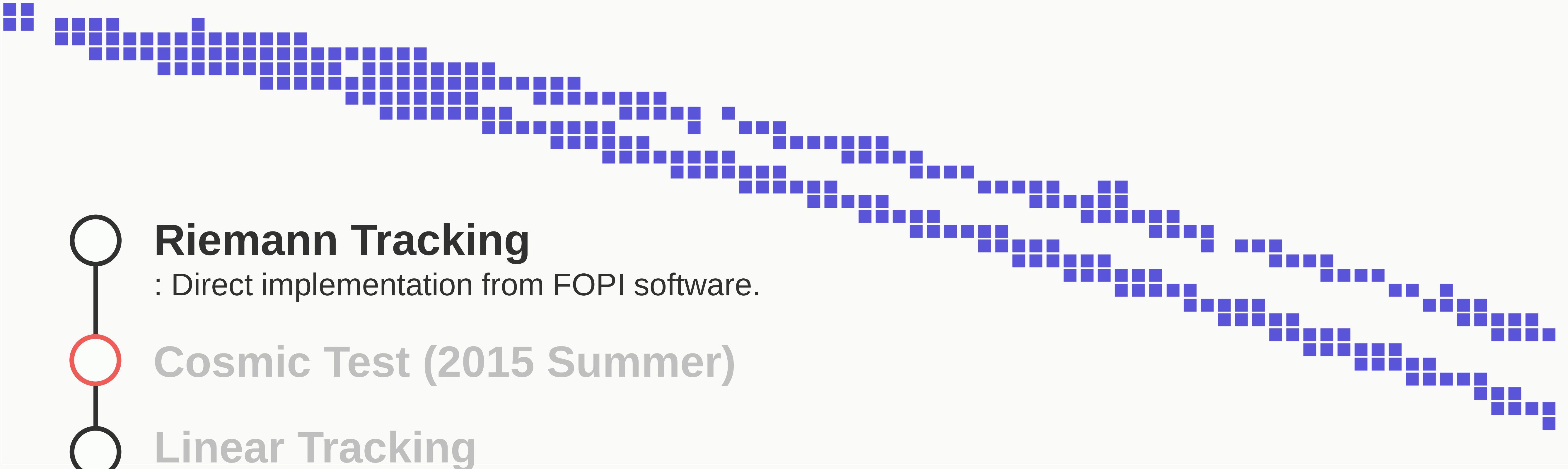
: Tracking inspired by linear tracking.

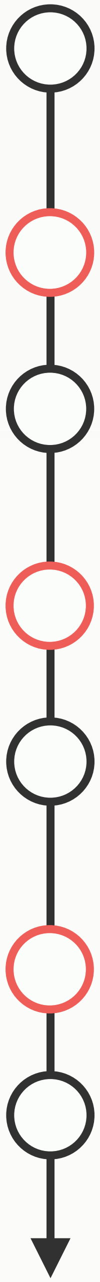
^{108}Sn, ^{132}Sn Run (April - May, 2016)

Helix Tracking

: Riemann fit + new hit searching algorithm. → **90 % track finding efficiency**

Riemann Tracking





Riemann Tracking
: Direct implementation from FOPI software.

Cosmic Test (2015 Summer)

Linear Tracking
: Tracking for linear track events (cosmic, commissioning)

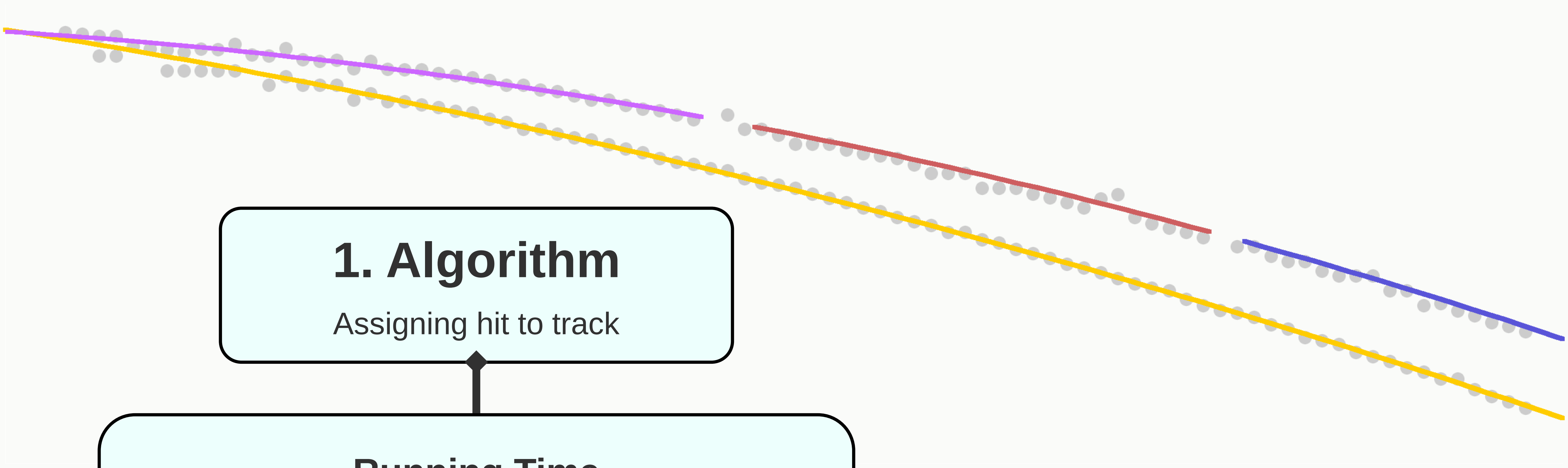
Commissioning Run (October, 2015)

Curve Tracking
: Tracking inspired by linear tracking.

^{108}Sn, ^{132}Sn Run (April - May, 2016)

Helix Tracking
: Riemann fit + new hit searching algorithm.

Riemann Tracking



1. Algorithm

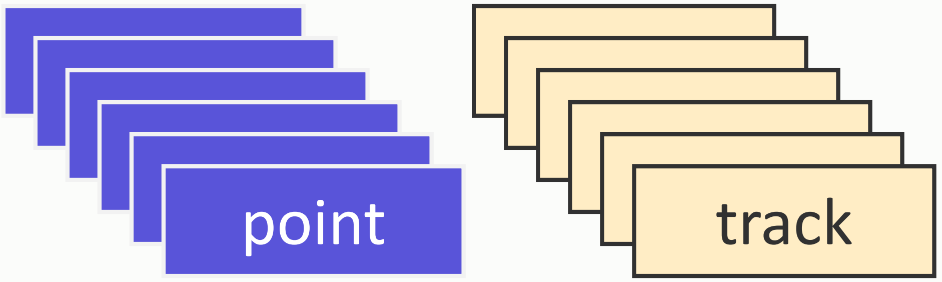
Assigning hit to track

Running Time

- Hit compared by number of created tracks.
- Tracks compared for merging by number of tracks.

Efficiency

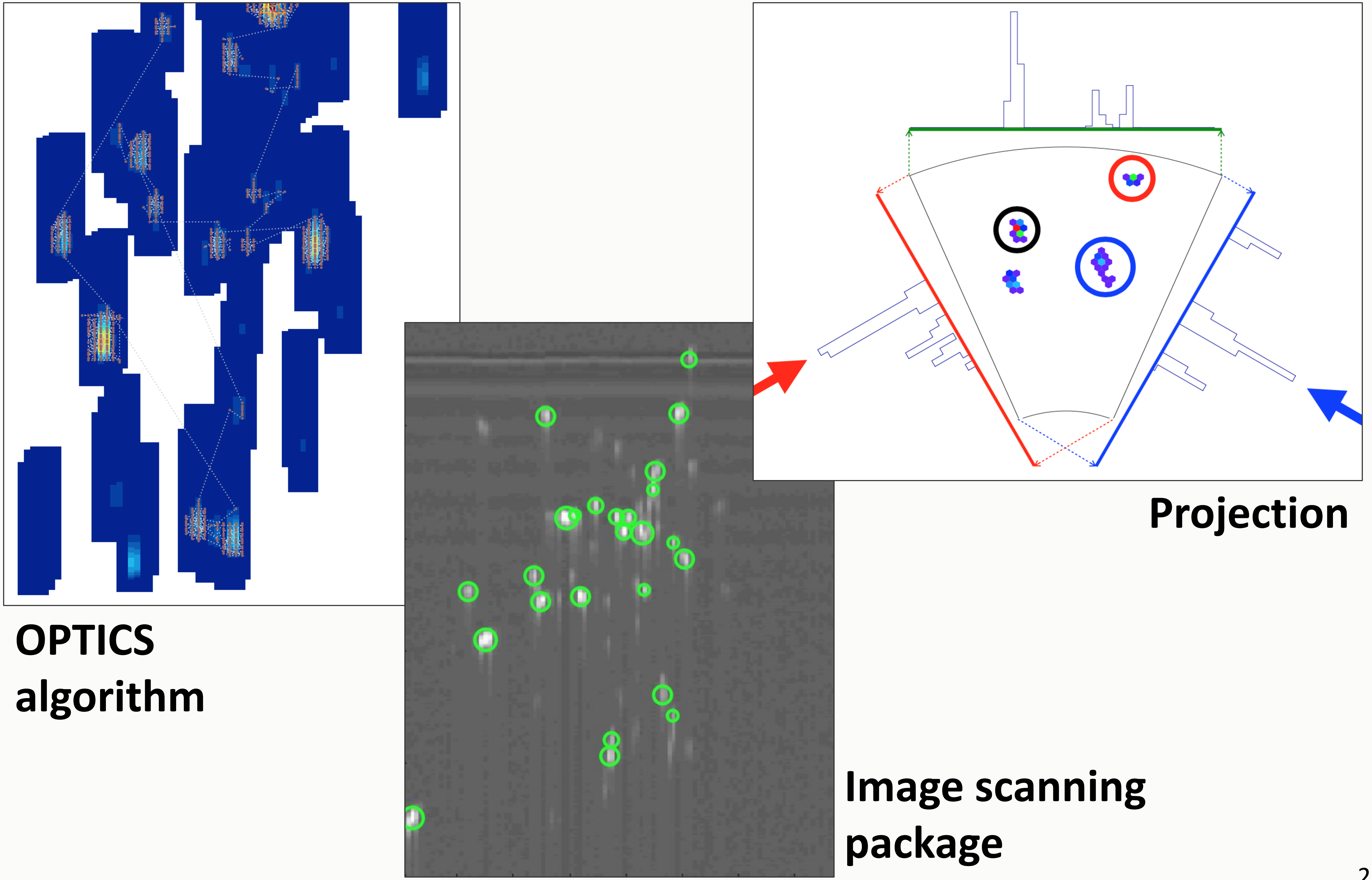
- Sensitive to noise.
- Sensitive to hit sorting → also creates broken tracks.
- Problem of pre-clustering.



2. Fast Helix Fit

Circle Fit (Riemann Fit) + Line Fit

Hit-Clustering



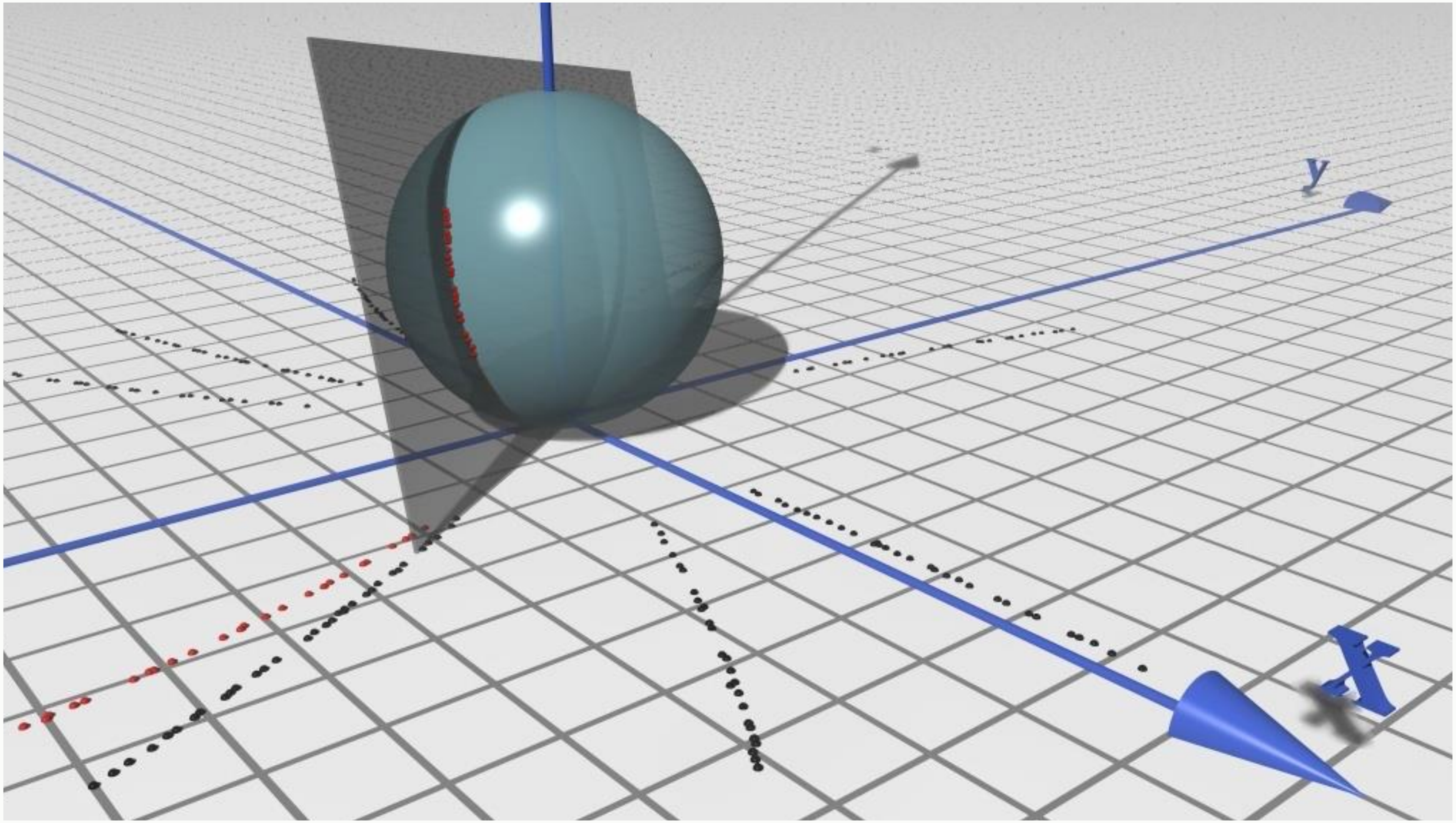
OPTICS algorithm

Projection

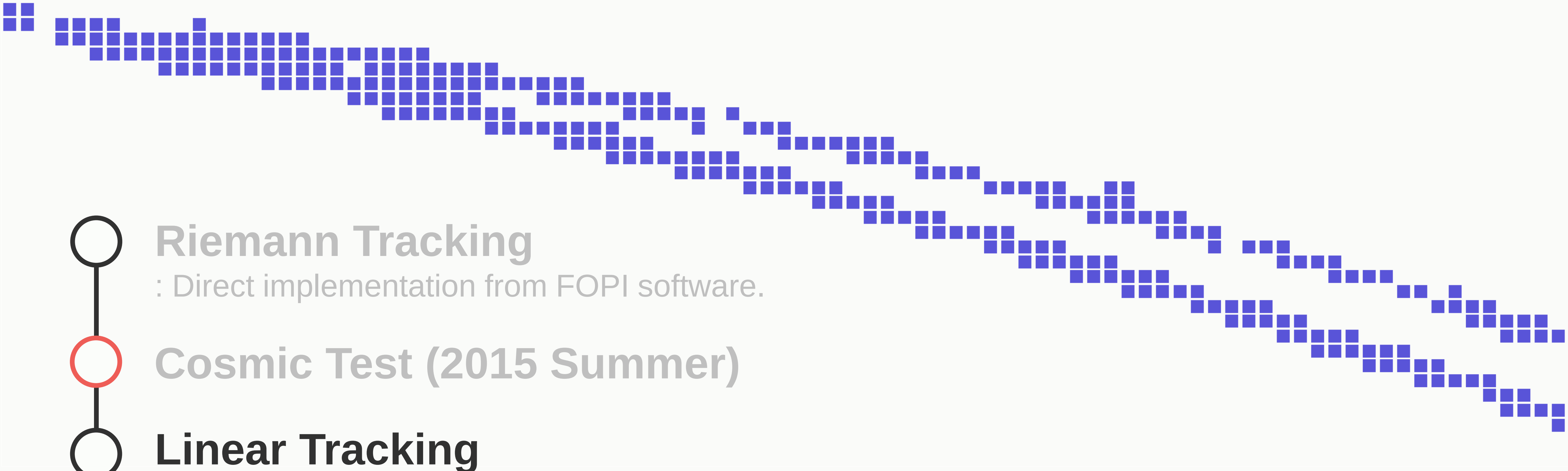
Image scanning package

Riemann Fit

1. Map track points from the pad plane to Riemann sphere surface.
2. Mapped points on the sphere form circle (plane) → fit plane (Eigen value equation)
3. Inverse map of circle on the Riemann sphere gives circle on the reference plane.



Linear Tracking



Riemann Tracking

: Direct implementation from FOPI software.

Cosmic Test (2015 Summer)

Linear Tracking

: Tracking for linear track events (cosmic, commissioning)

Commissioning Run (October, 2015)

Curve Tracking

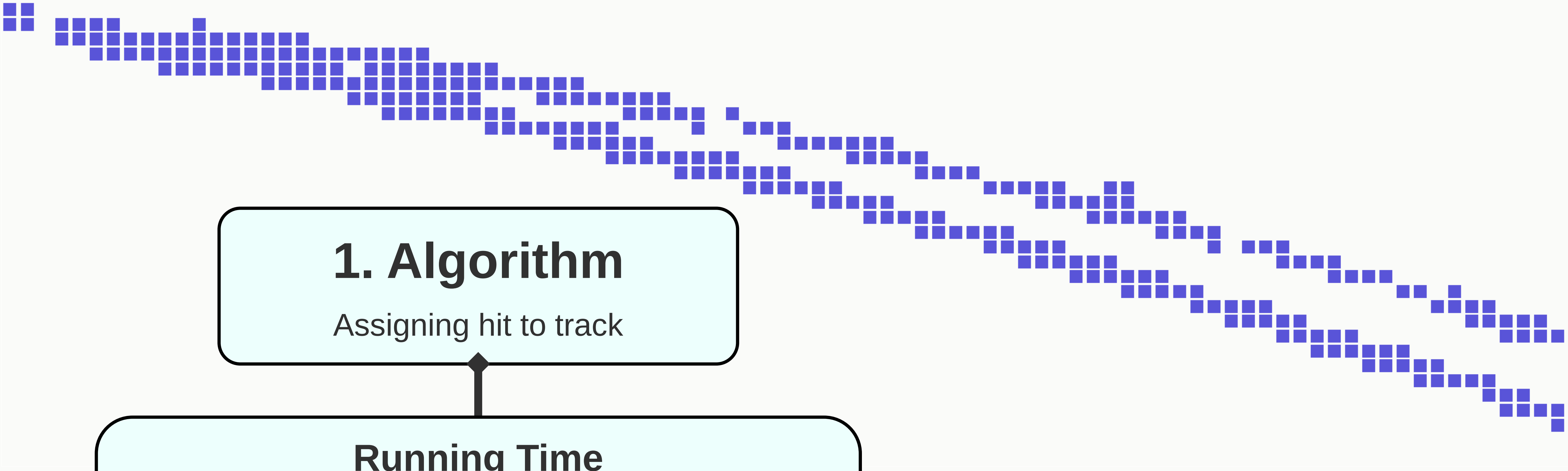
: Tracking inspired by linear tracking.

^{108}Sn, ^{132}Sn Run (April - May, 2016)

Helix Tracking

: Riemann fit + new hit searching algorithm.

Linear Tracking



1. Algorithm

Assigning hit to track

Running Time

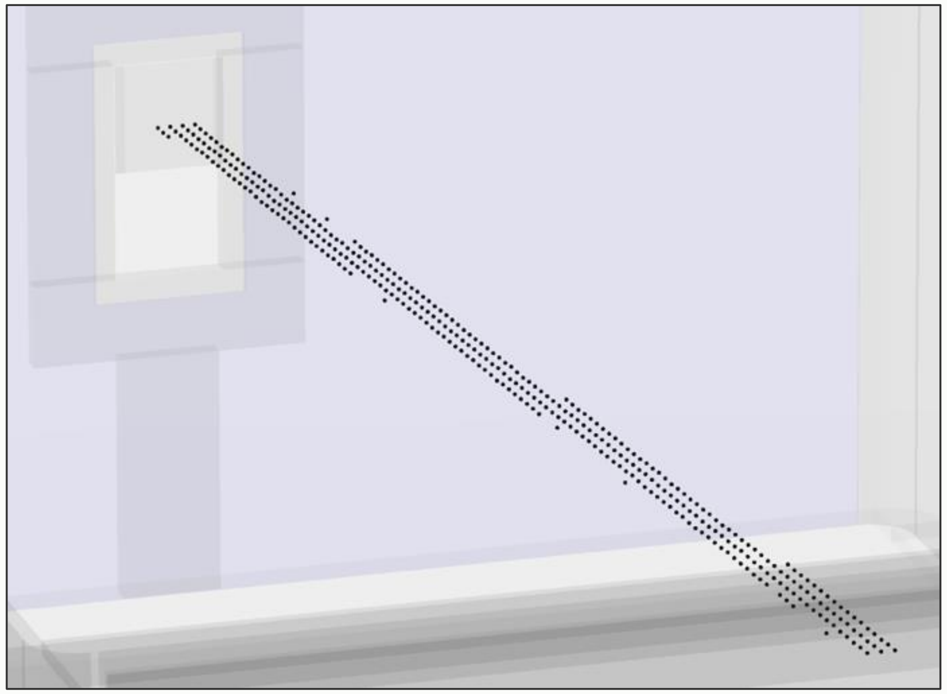
- Hit compared by number of created tracks.
- Tracks compared for merging by number of tracks.

Efficiency

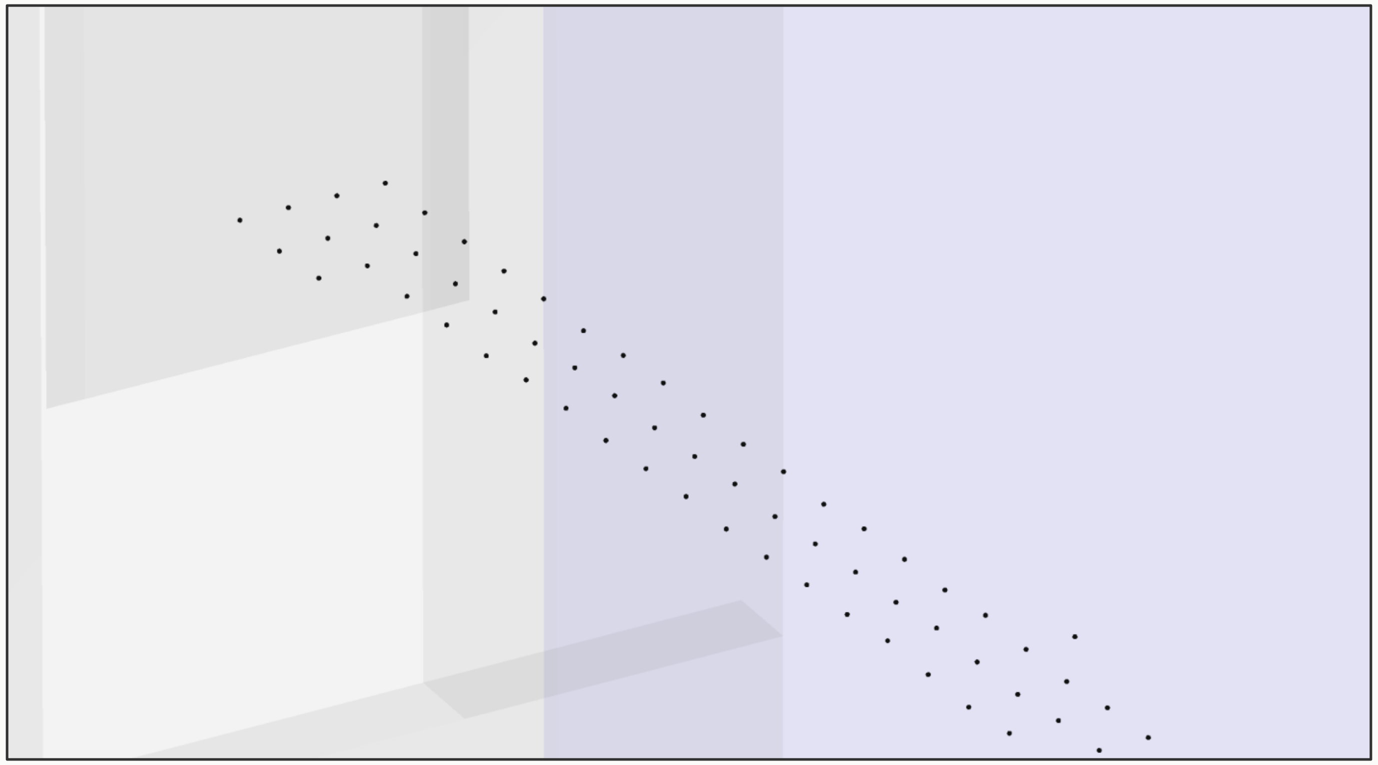
- Sensitive to noise.
- Sensitive to hit sorting → also creates broken tracks.
- Improved by using plane formation of tracks.

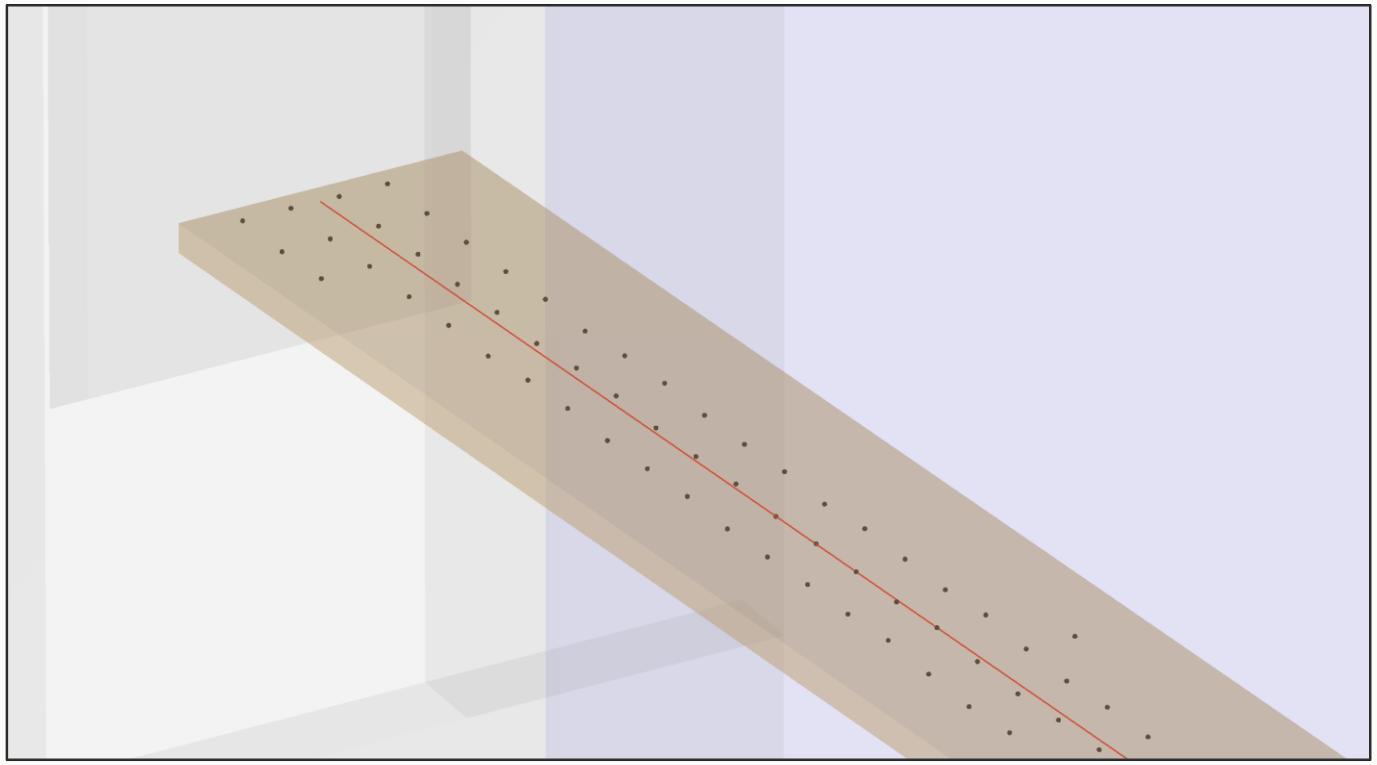
2. Linear Track Fit

Plane Fit, Line Fit

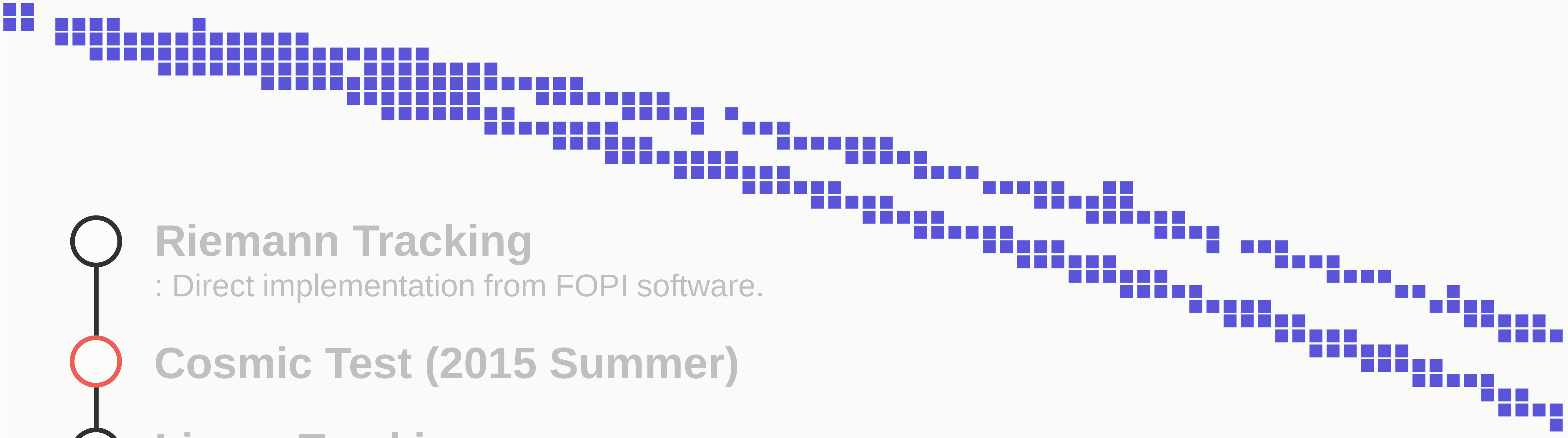


Linear Tracking





Curve Tracking



Riemann Tracking

: Direct implementation from FOPI software.

Cosmic Test (2015 Summer)

Linear Tracking

: Tracking for linear track events (cosmic, commissioning)

Commissioning Run (October, 2015)

Curve Tracking

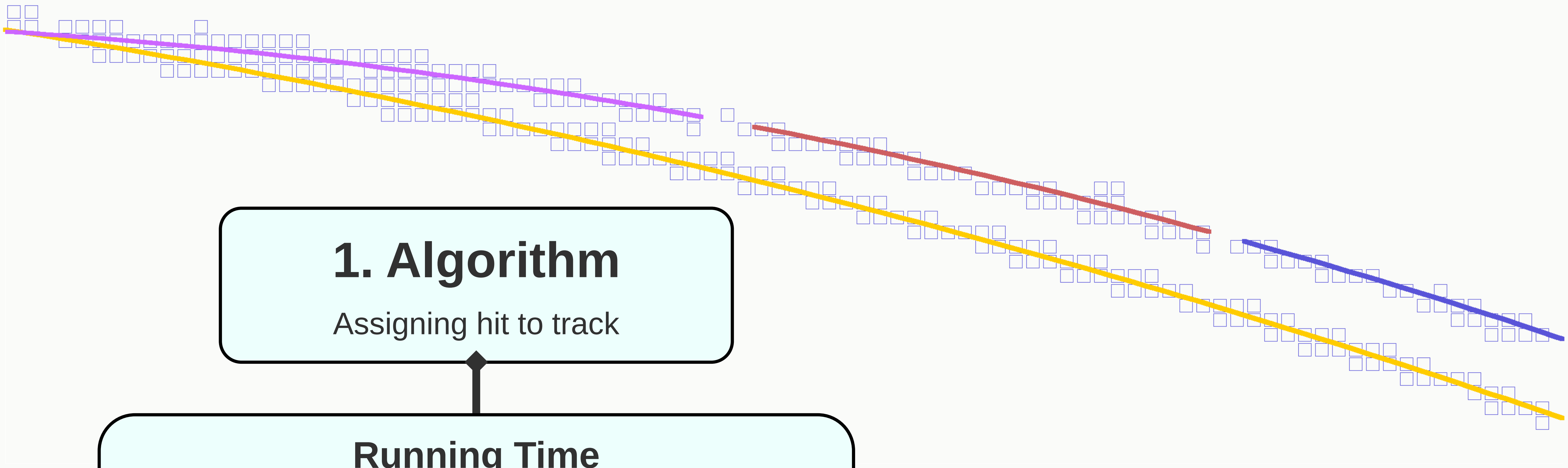
: Tracking inspired by linear tracking.

^{108}Sn, ^{132}Sn Run (April - May, 2016)

Helix Tracking

: Riemann fit + new hit searching algorithm.

Curve Tracking



1. Algorithm

Assigning hit to track

Running Time

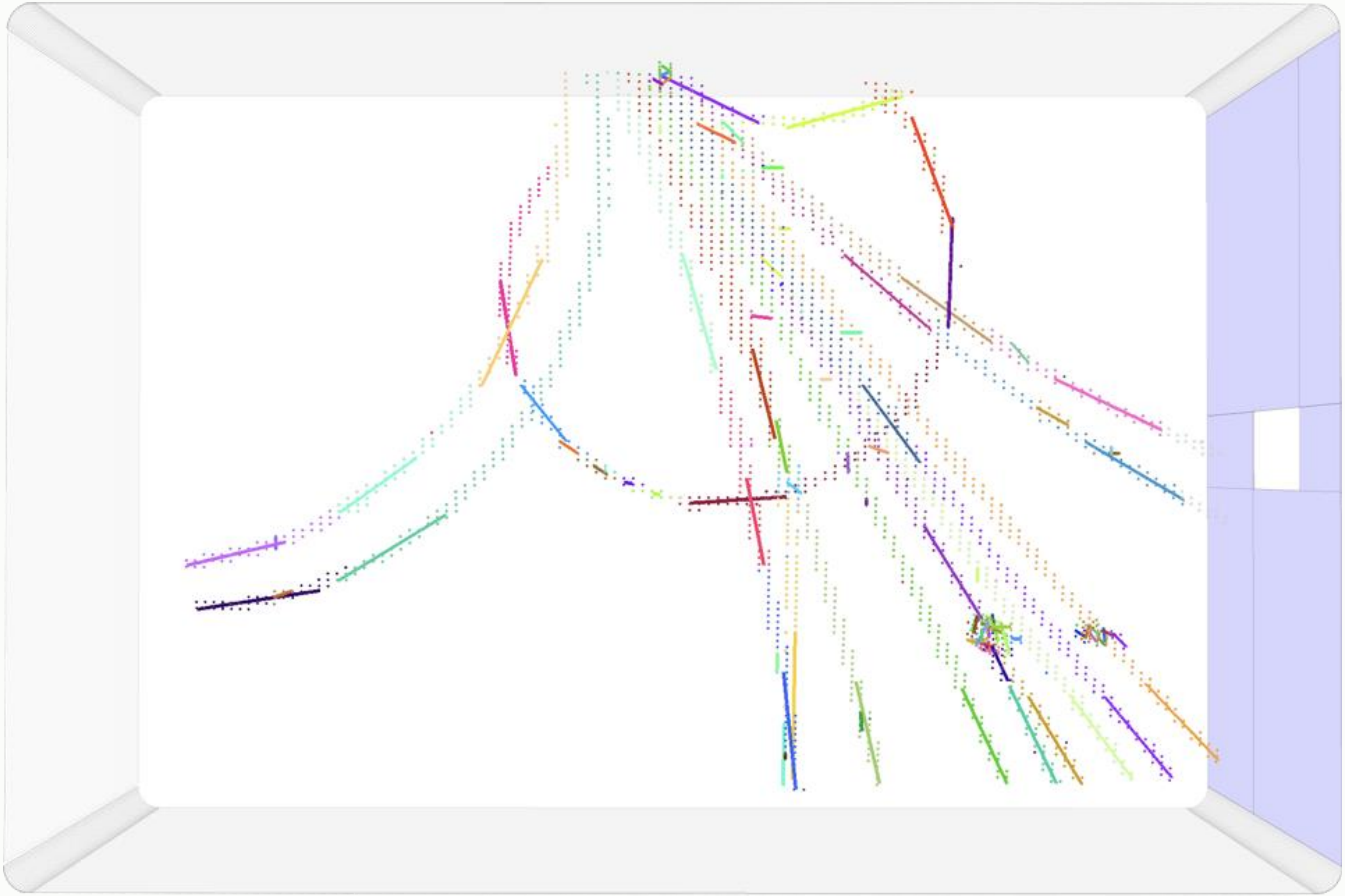
- Hit compared by number of created tracks.
- Tracks compared for merging by number of tracks.

Efficiency

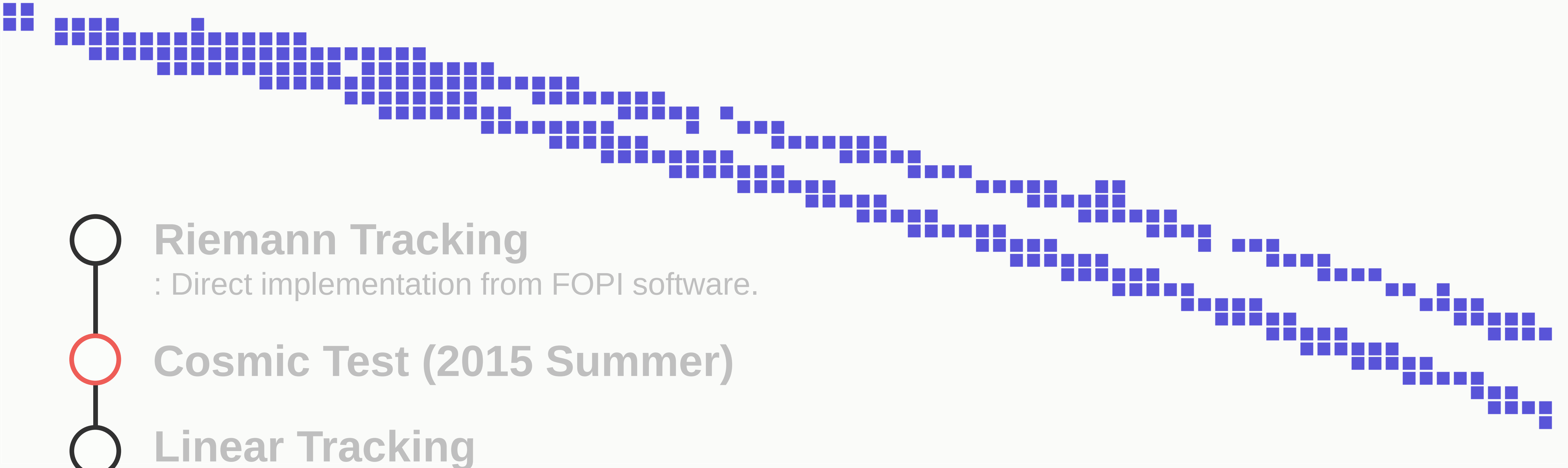
- Sensitive to hit sorting → also creates broken tracks.
- Improved by using plane formation of tracks.
- Not looking at whole part of the track

2. Linear Track Fit

Local Plane Fit, Line Fit



Helix Tracking



Riemann Tracking

: Direct implementation from FOPI software.

Cosmic Test (2015 Summer)

Linear Tracking

: Tracking for linear track events (cosmic, commissioning)

Commissioning Run (October, 2015)

Curve Tracking

: Tracking inspired by linear tracking.

^{108}Sn, ^{132}Sn Run (April - May, 2016)

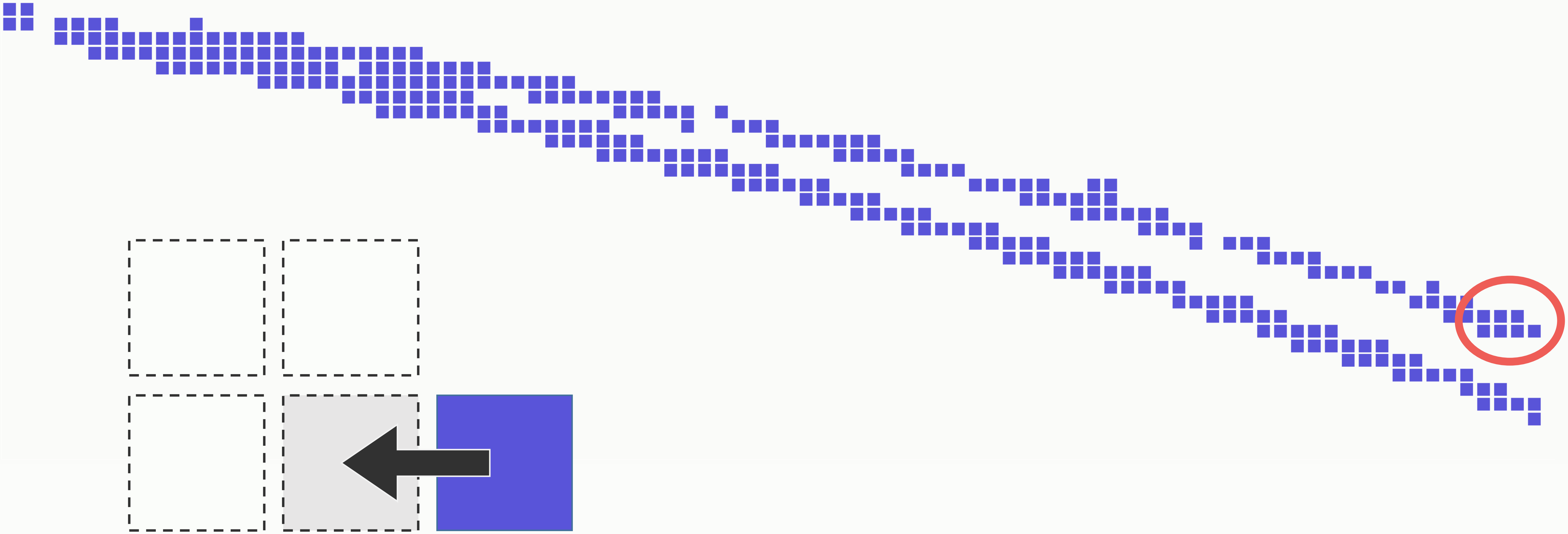
Helix Tracking

: Riemann fit + new hit searching algorithm.

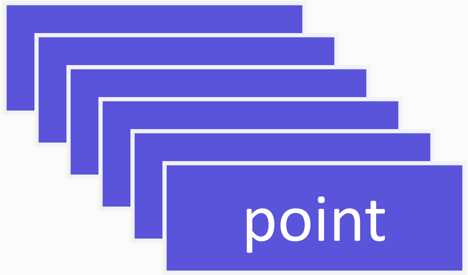
Until here,

- Too much dependence on parameters
- Too much dependence on sorting

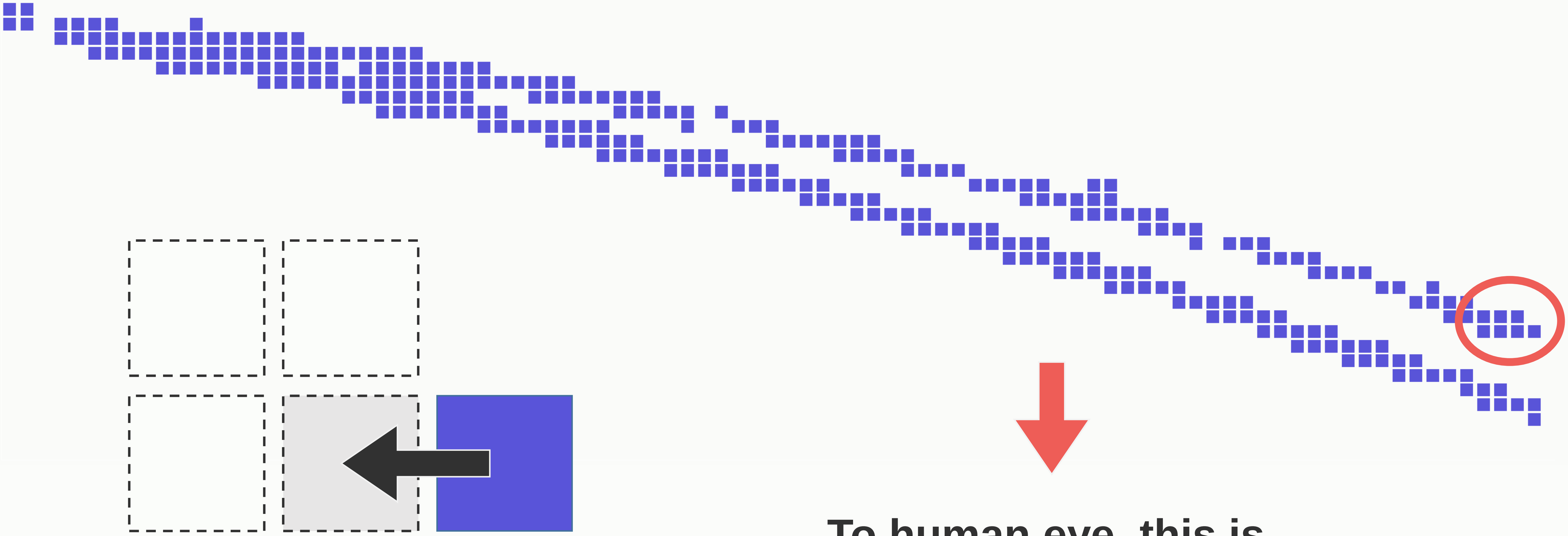
Helix Tracking



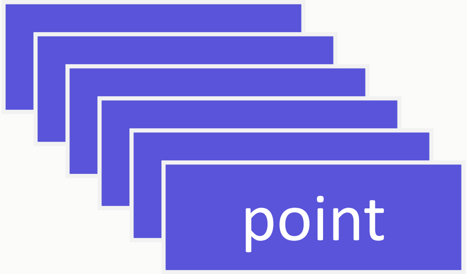
Need ~300 comparisons before closest hit is found



Helix Tracking



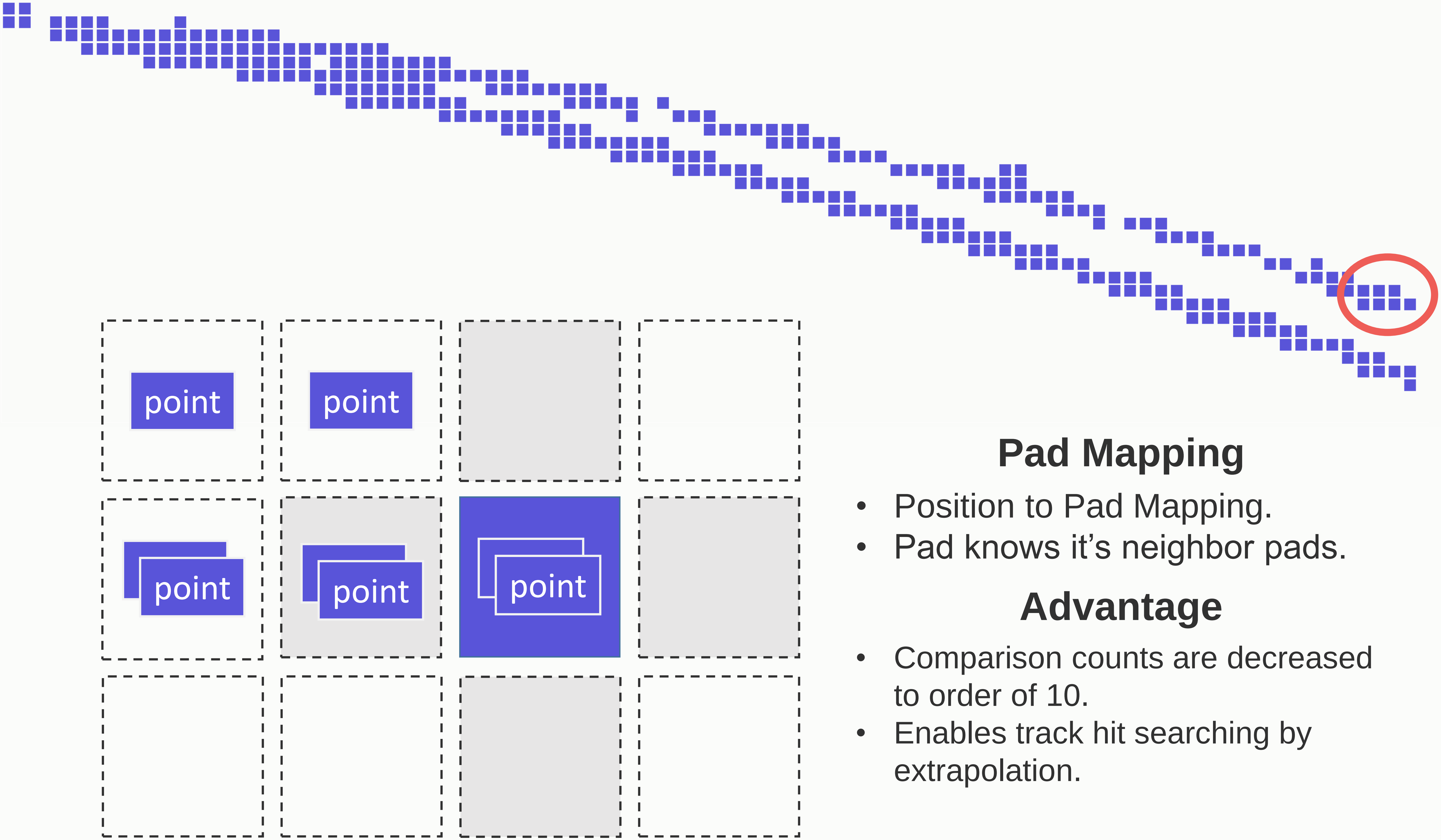
Need ~300 comparisons before closest hit is found



To human eye, this is 2D space map data

We can find close hit
We can draw smooth line.

Helix Tracking



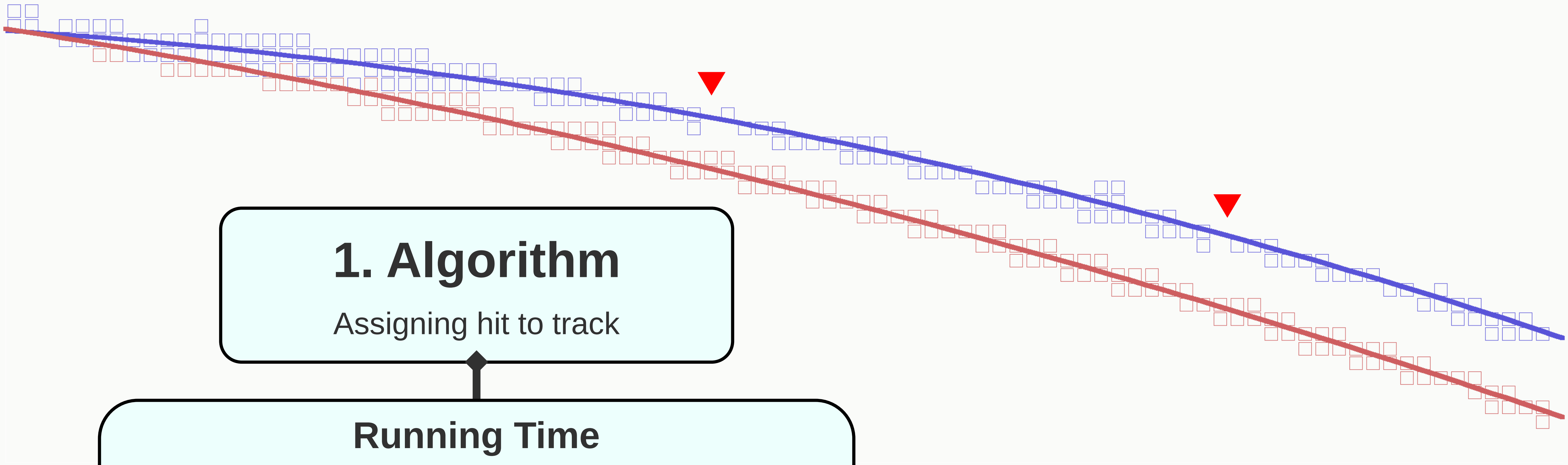
Pad Mapping

- Position to Pad Mapping.
- Pad knows it’s neighbor pads.

Advantage

- Comparison counts are decreased to order of 10.
- Enables track hit searching by extrapolation.

Helix Tracking



1. Algorithm

Assigning hit to track

Running Time

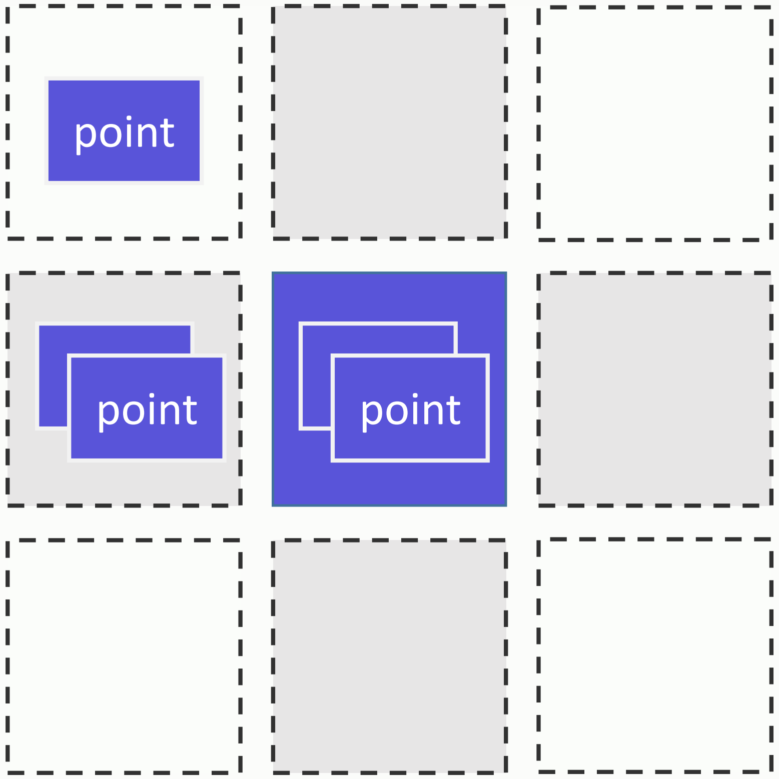
- Comparison counts are in order of 10 times for adding one hit to track
- One track is fully built before the other starts

Efficiency

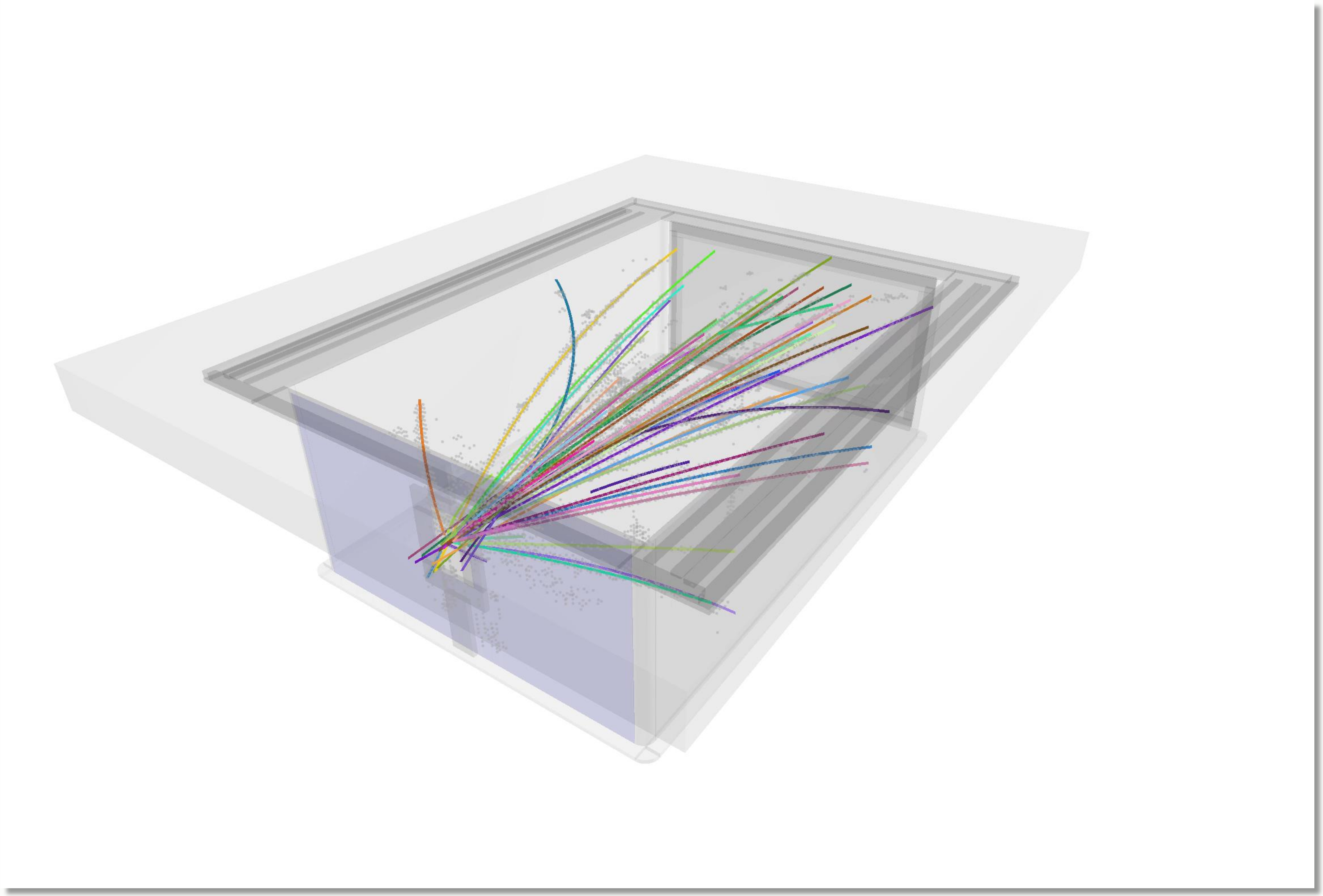
- No need of hit sorting
- Depend on pad plane geometry and pad shape.
- Track extrapolation is used to link broken part.

2. Fast Helix Fit

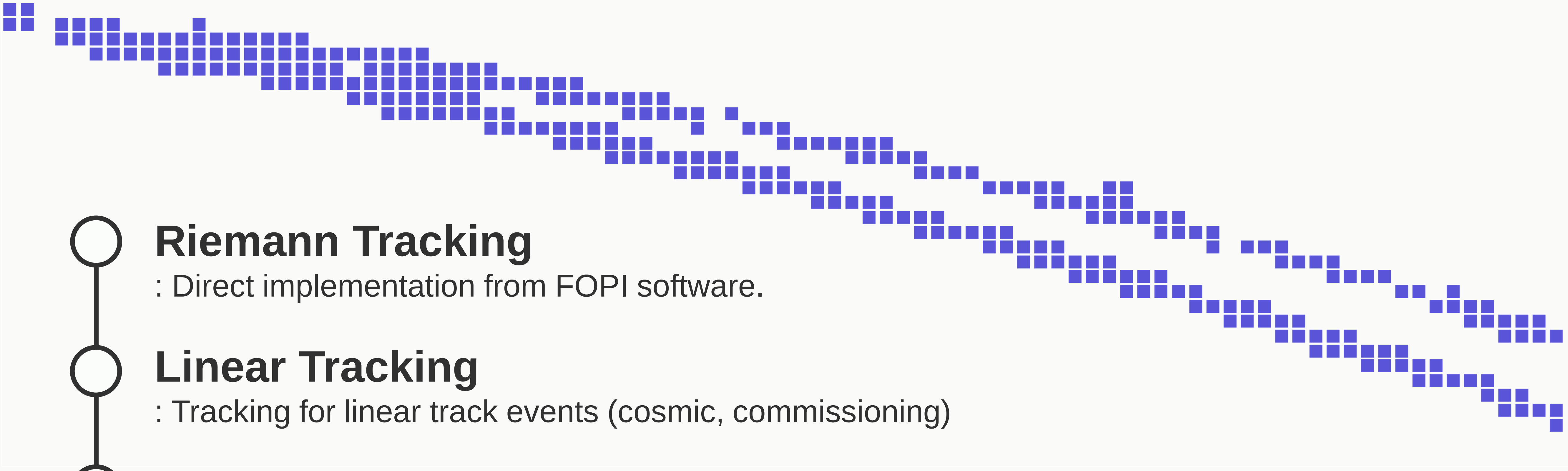
Circle Fit (Riemann Fit) + Line Fit



Helix Tracking



Summary



Riemann Tracking

: Direct implementation from FOPI software.

Linear Tracking

: Tracking for linear track events (cosmic, commissioning)

Curve Tracking

: Tracking inspired by linear tracking.

Helix Tracking

: Riemann fit + new hit searching algorithm. → **90 % track finding efficiency**

- **Helix Tracking + Curve Tracking?**
- **Machine Learning (Neural Network)**