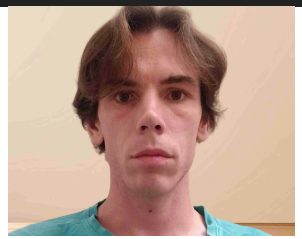
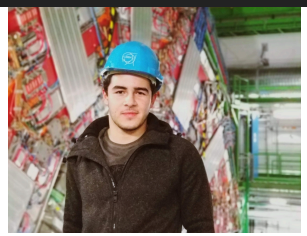
Tracking & Vertexing


Emery Nibigira


Leonardo Giannini


Tetiana Mazurets


Hichem Bouchamaoui


Ray Wynne

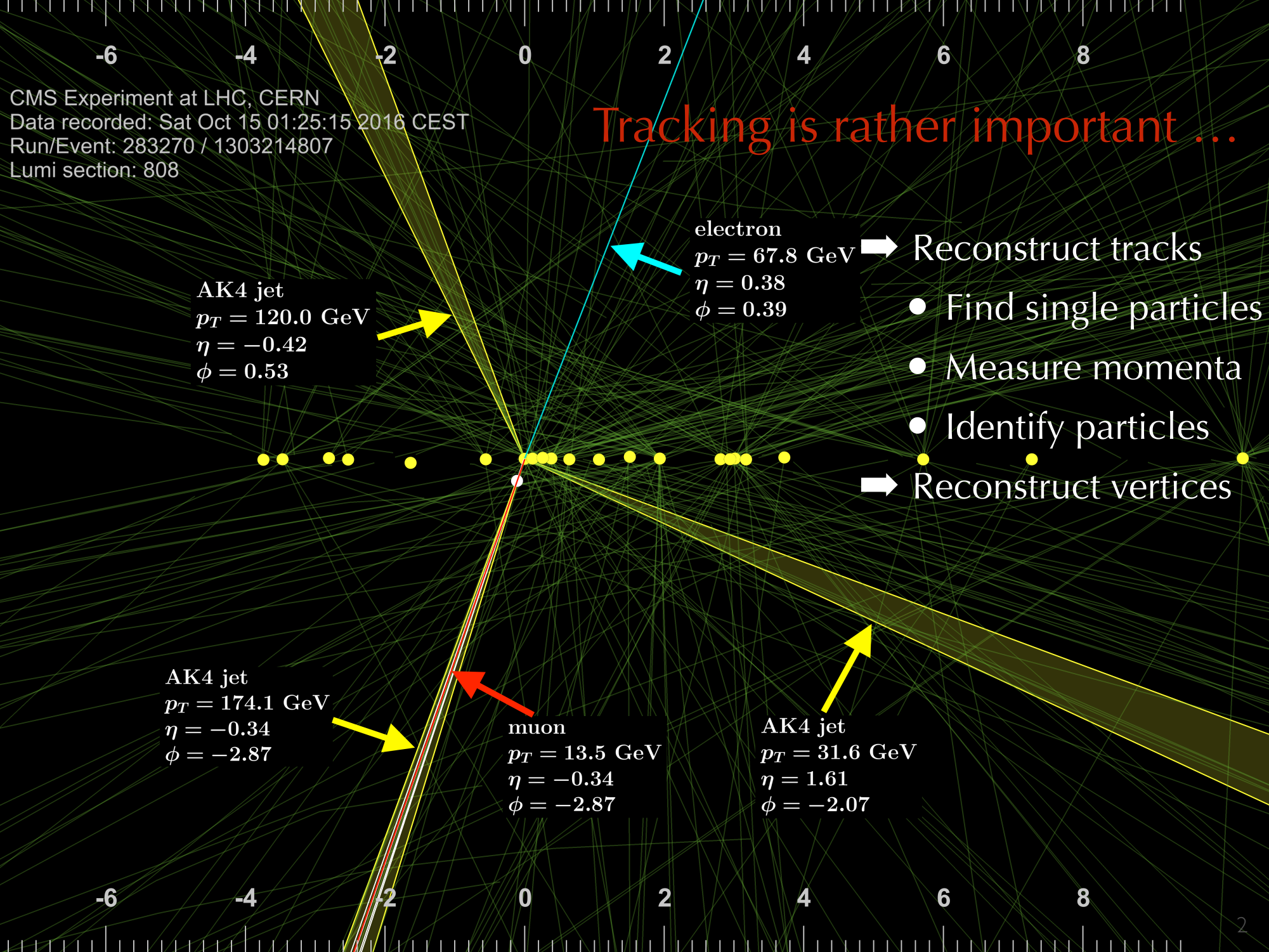

Aidan Grummer

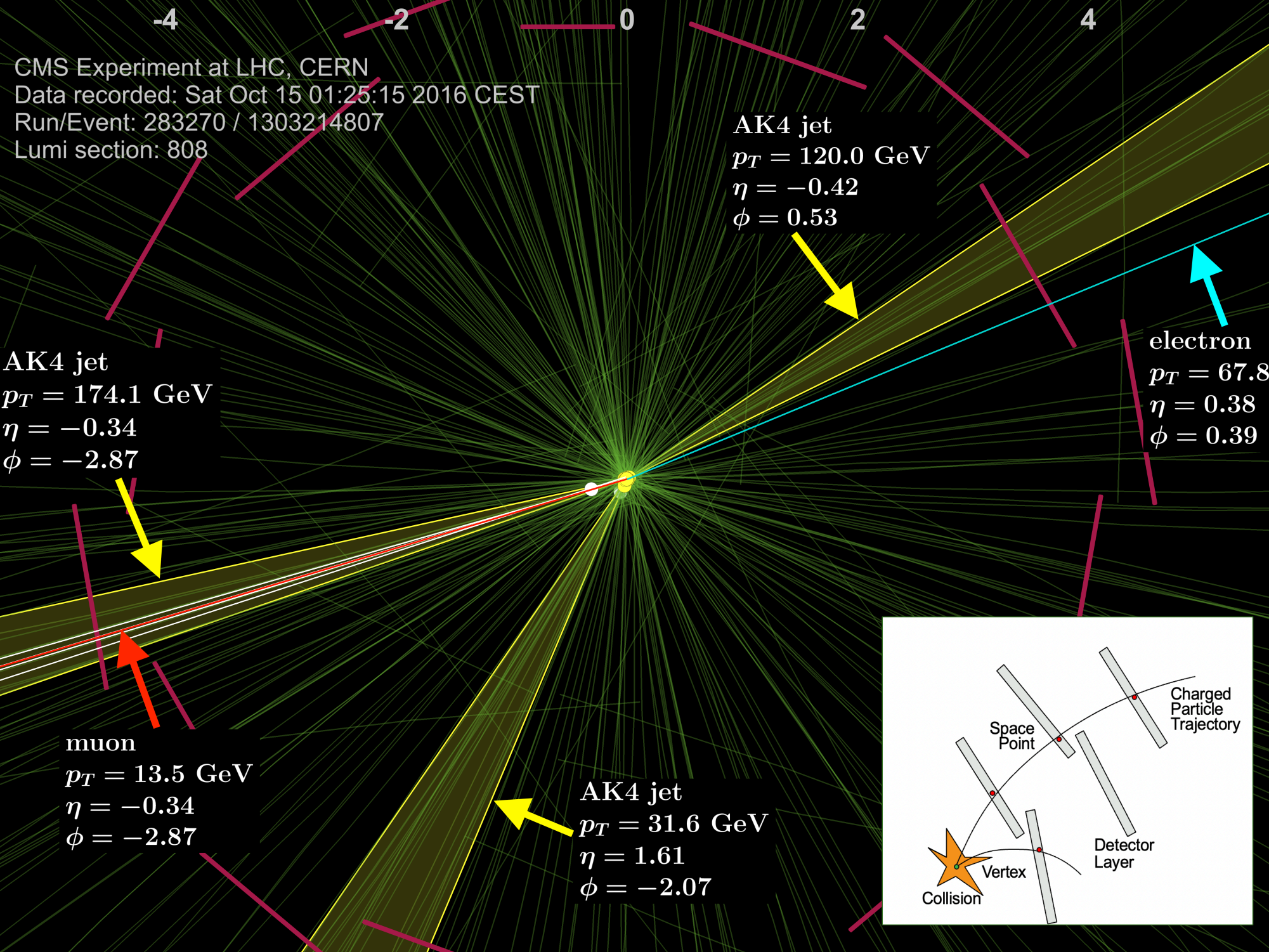
CMS Data Analysis School 2025

Fermilab

Jan 13-17, 2025

slides thankfully borrowed and updated from Kirill Skovpen


-6
-4
-2
0
2
4
6
8
CMS Experiment at LHC, CERN
Data recorded: Sat Oct 15 01:25:15 2016 CEST
Run/Event: 283270 / 1303214807
Lumi section: 808
Tracking is rather important …
electron
$p_T = 67.8$ GeV
$\eta = 0.38$
$\phi = 0.39$
AK4 jet
$p_T = 120.0$ GeV
$\eta = -0.42$
$\phi = 0.53$
Reconstruct tracks
Find single particles
Measure momenta
Identify particles
Reconstruct vertices
AK4 jet
$p_T = 174.1$ GeV
$\eta = -0.34$
$\phi = -2.87$
muon
$p_T = 13.5$ GeV
$\eta = -0.34$
$\phi = -2.87$
AK4 jet
$p_T = 31.6$ GeV
$\eta = 1.61$
$\phi = -2.07$


-4
-2
0
2
4
CMS Experiment at LHC, CERN
Data recorded: Sat Oct 15 01:25:15 2016 CEST
Run/Event: 283270 / 1303214807
Lumi section: 808
AK4 jet
$p_T = 120.0$ GeV
$\eta = -0.42$
$\phi = 0.53$
electron
$p_T = 67.8$
$\eta = 0.38$
$\phi = 0.39$
AK4 jet
$p_T = 174.1$ GeV
$\eta = -0.34$
$\phi = -2.87$
muon
$p_T = 13.5$ GeV
$\eta = -0.34$
$\phi = -2.87$
AK4 jet
$p_T = 31.6$ GeV
$\eta = 1.61$
$\phi = -2.07$
Charged Particle Trajectory
Space Point
Detector Layer
Vertex
Collision

Menu for today

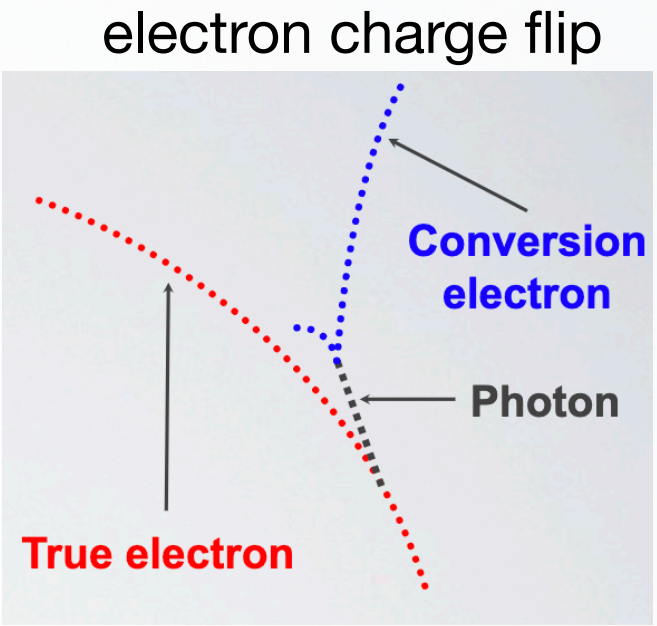
- ➡ Get to know the CMS Silicon Tracker and **how the tracks and vertices are reconstructed**
- ➡ Learn how to **assess track information** for analysis purposes in different data formats
- ➡ Use track information to understand the **particle interactions** in an event
- ➡ Use track information to measure the **tracking efficiency**

Tracking is everything

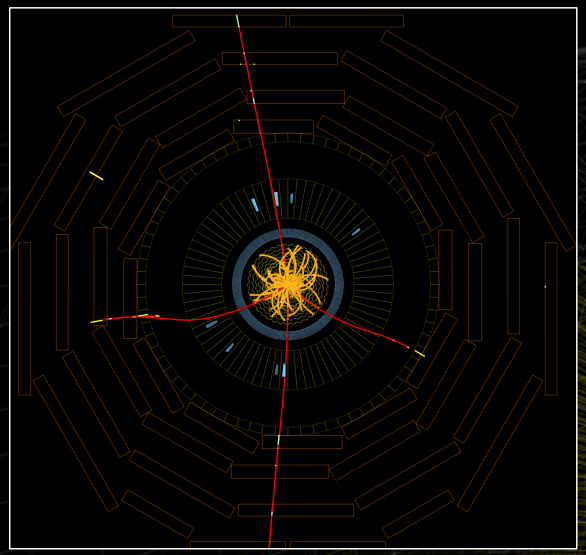
- The reconstruction of charged particle tracks and vertices is **fundamental** to the reconstruction of every type of physics event in CMS:
  - Directly used in the **reconstruction** of **charged hadrons**, **electrons**, and **muons**
  - Needed to distinguish **charged** and **neutral hadrons** as well as **electrons** from **photons**
  - Crucial ingredient for **higher level objects** like (b-tagged) jets or taus, missing transverse momentum (MET)
  - Association of tracks to vertices needed to **distinguish particles from the hard interaction** and **pileup vertices**
  - Secondary vertices crucial to **track the decay chains of particles**

… and crucial for analysis

- Good resolution and high efficiency are crucial for physics analyses
- Example: Charge mismeasurements of electrons, can happen through various effects
  - Small curvature for high p_T electrons
  - Photon conversions from bremsstrahlung
- Good tracking and sensible track selections can substantially reduce backgrounds arising from this effect

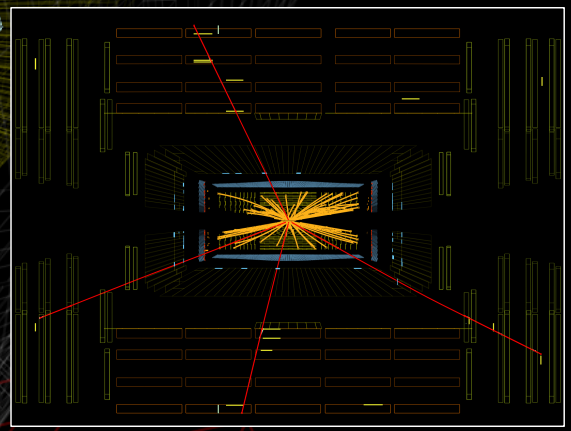


Tracking is challenging



Tracker (Pixel & Strip)

Muon (hits in Muon stations)



- On average, 30 charged particles within the Tracker acceptance per proton-proton collision and ≈ 25-60 interactions per event → O(1000) charged particles per event are needed to be reconstructed

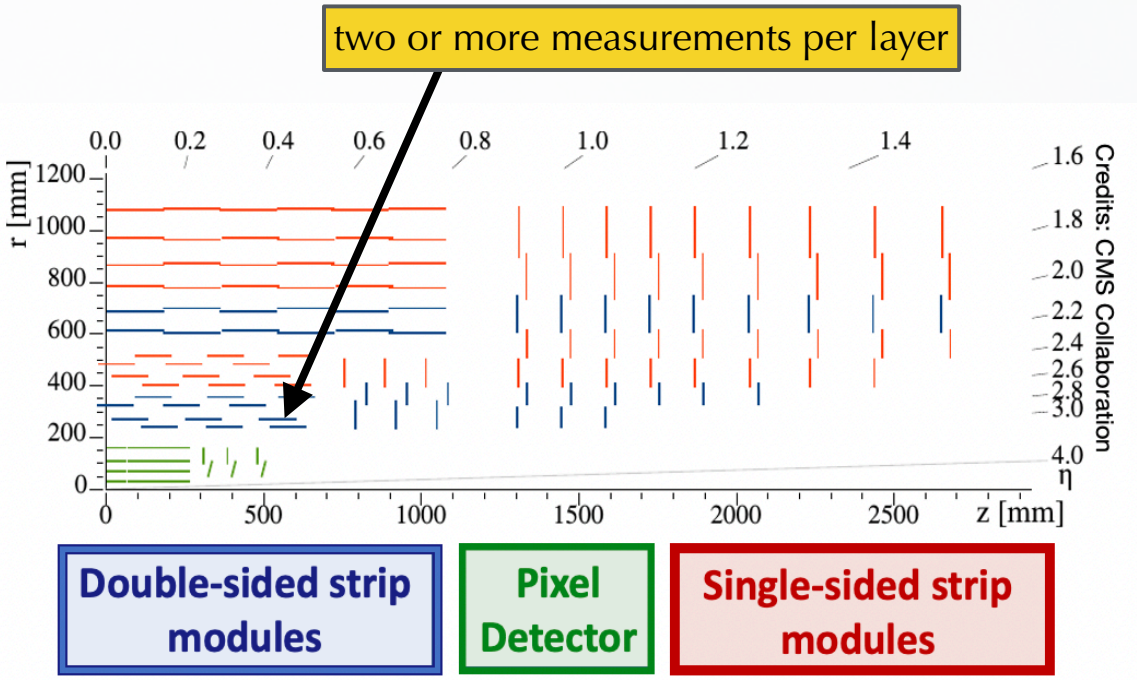
Tracker

15 hits per track on average

$\sigma(p_T)/p_T \sim$ **1-2%** @100 GeV

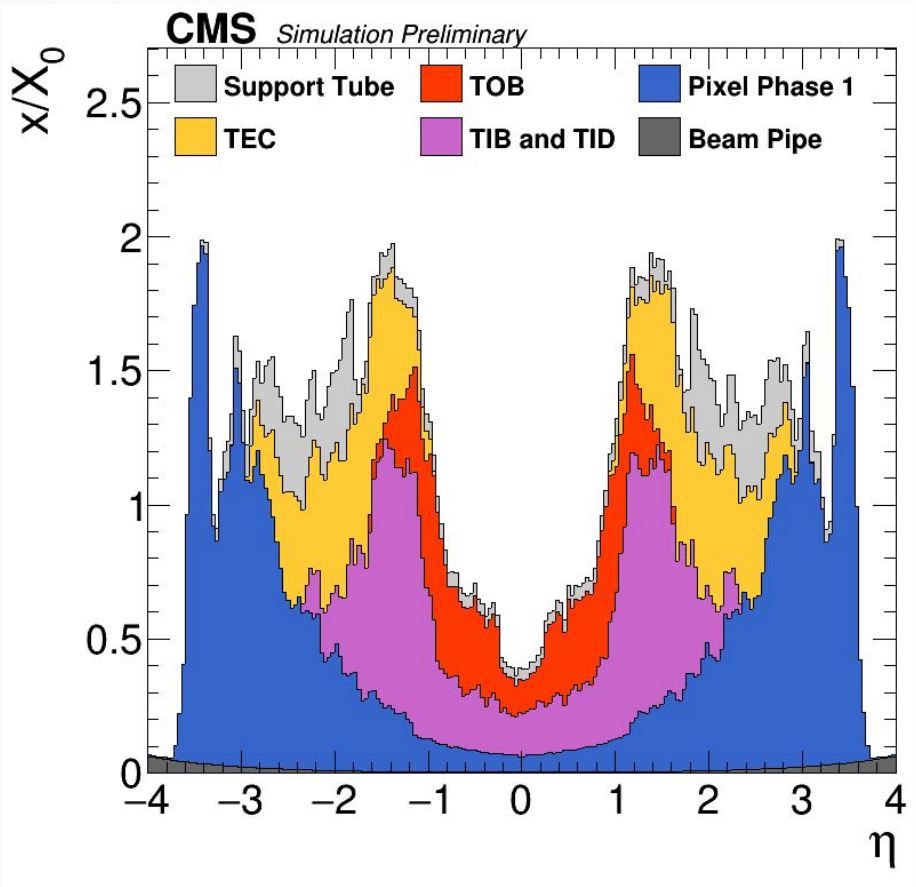
$\sigma(IP) \sim$ **10-20 µm** @10-100 GeV

- **Position information** from finely segmented silicon sensors:
  - ➡ Record the **path of charged particles**
  - ➡ Measure **momentum** from bending radius in the 3.8 T magnetic field
  - ➡ Reconstruct **primary** and **secondary vertices**
- **Requirements**:
  - **High resolution** & **low occupancy** to resolve and isolate individual tracks and reconstruct vertices
  - **Finer granularity** close to the interaction point due to high particle density
  - **High rate capability** for fast charge collection and readout electronics for expected high rates
  - **Low material budget** to minimize multiple scattering effects
  - **Radiation hardness** for operation in the area with highest particle flux

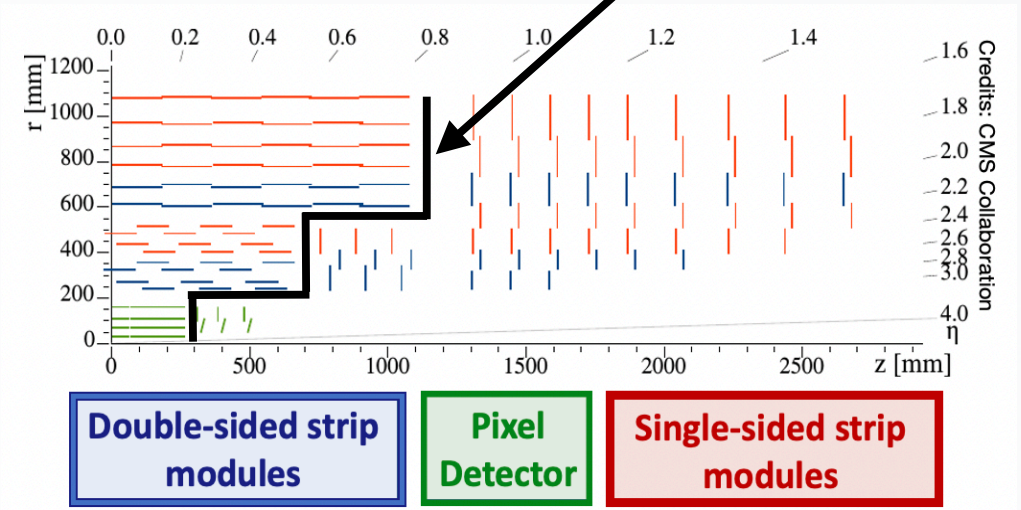


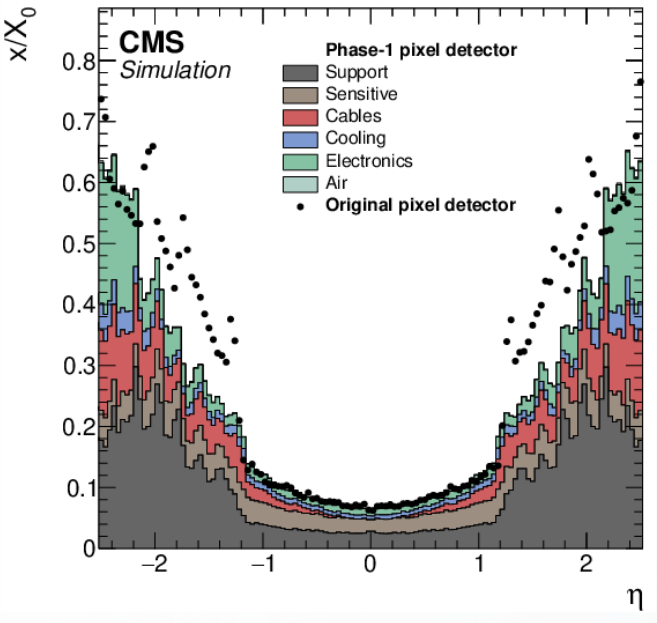
Material budget

- Most of the services are located in the **barrel-endcap transition** region
- **Amount of material** crossed by a primary track increases due to geometrical effect as $l = h/\sin(\theta)$



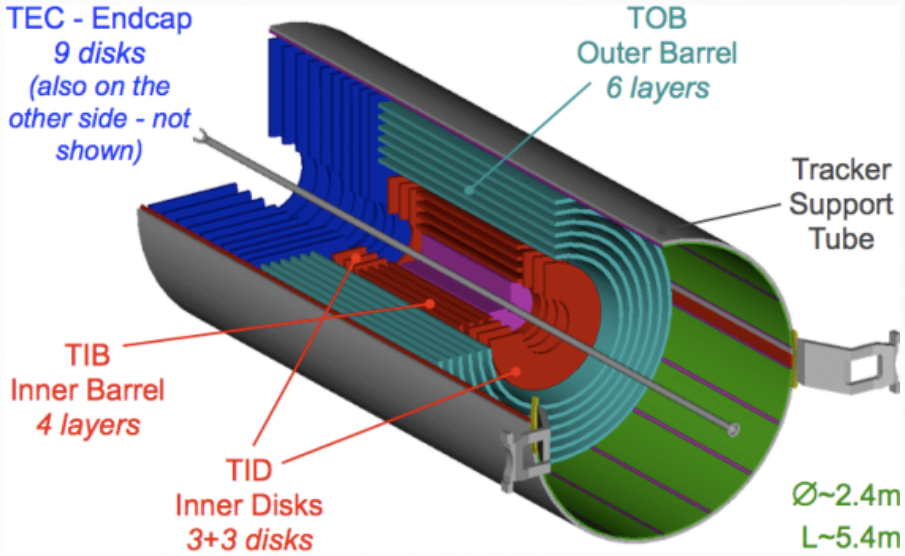
regions of concentration of supporting services



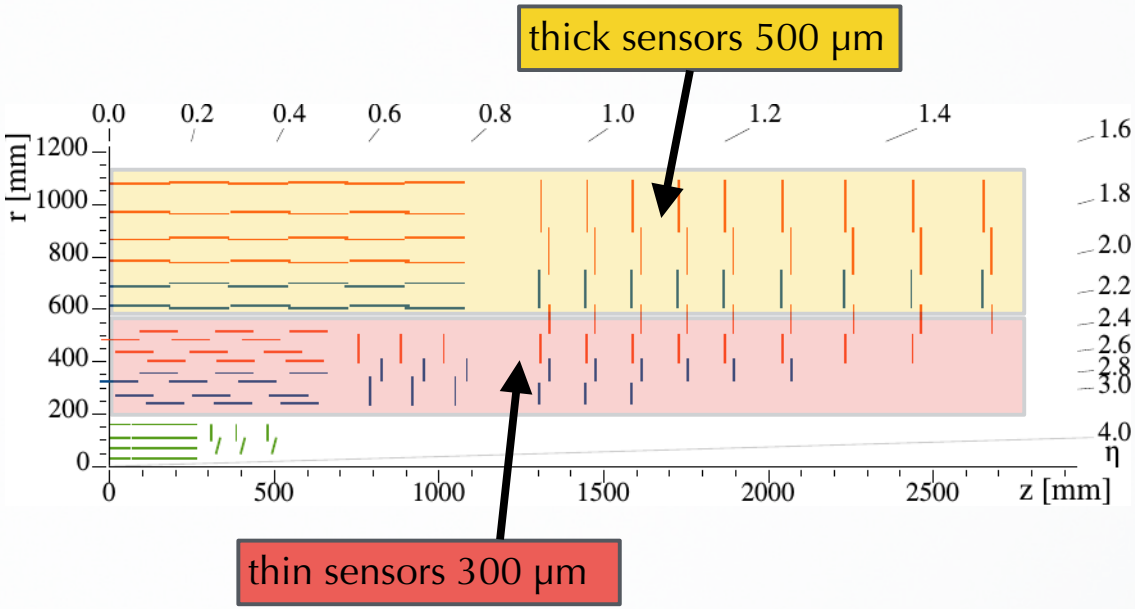


Silicon strips

- **O(10) million** strips
- **O(200) m²** of sensors
- **Hit resolution**: (10,40)x(230,530) μm
- **Occupancy**: 1-3%
- **Coverage** up to $|\eta| < 2.5$
- **12 hits** per track on average

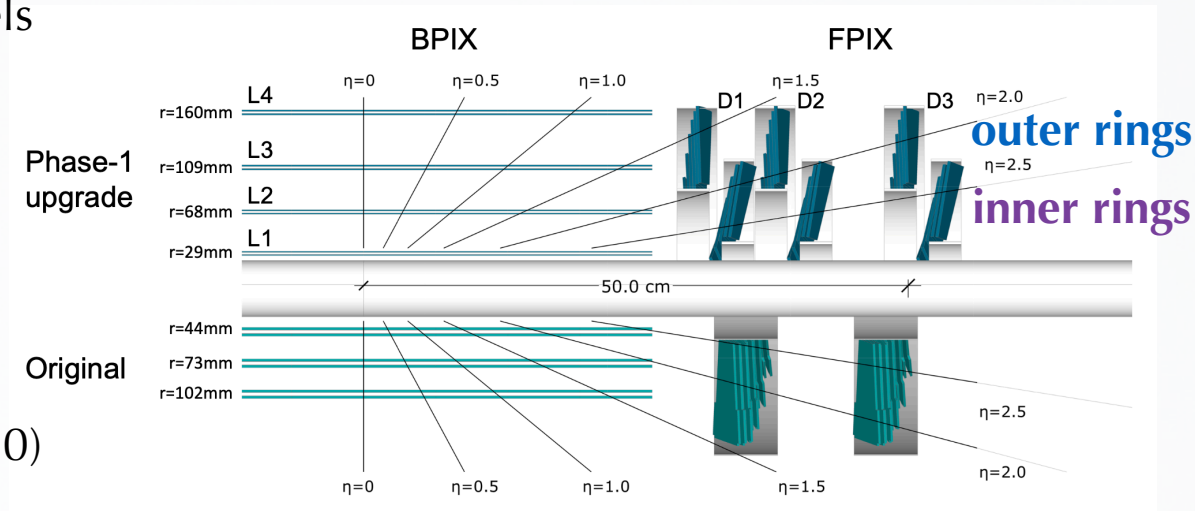


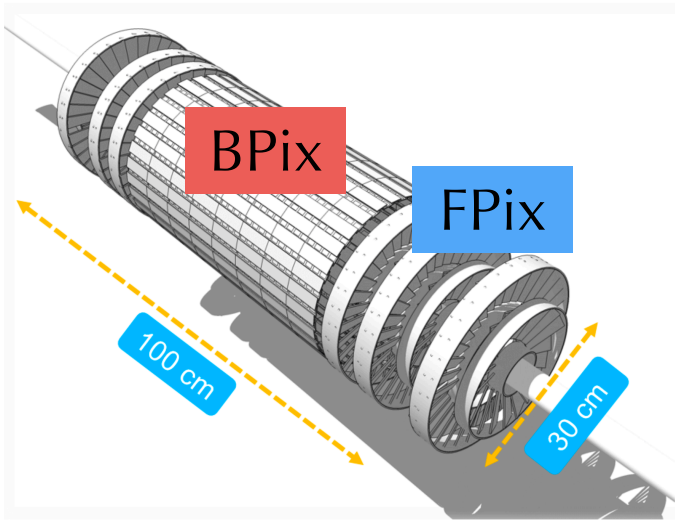
- **Sub detectors**:
  - Inner Barrel (**TIB**): 4
  - Inner Disks (**TID**): 3 (x2)
  - Outer Barrel (**TOB**): 6
  - Endcap (**TEC**): 9 (x2)



Silicon pixels

- **127 million** of **100x150 μm²** pixels
- **Hit resolution**: 10x(20,40) μm
- **Occupancy**: 0.1%
- **Layer position** [cm]
  BPix: 2.9, 6.8, 10.9, 16.0
  FPix: 29.1, 39.6, 51.6
- **Coverage** up to $|\eta| < 2.5$ (even 3.0)
- **4 hits** per track on average
- **High segmentation** with high quality seeds for offline tracking

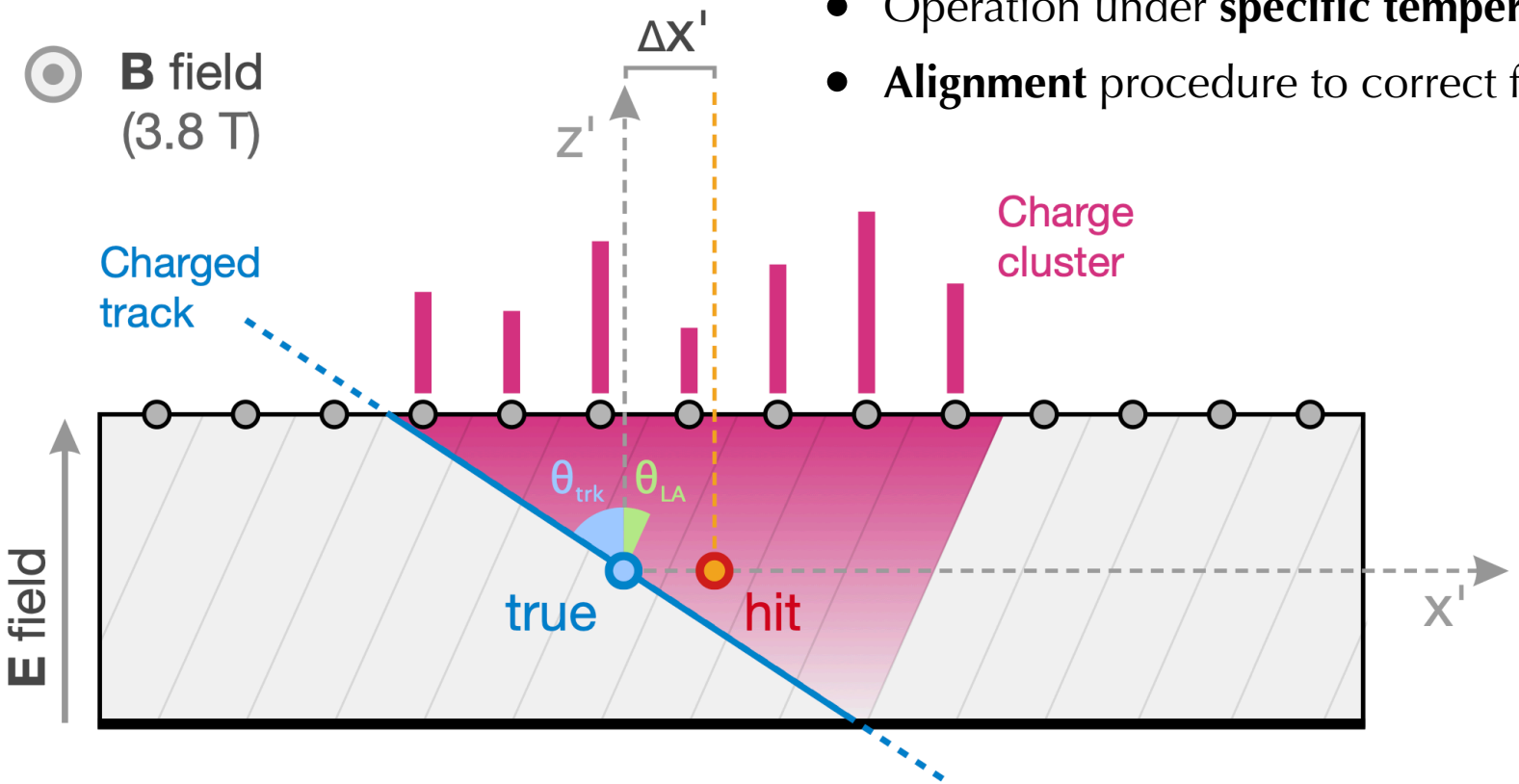




- **Since 2017**, **one additional tracking point** in both barrel and forward regions → **4-hit seeds and lower fake rate (fake track = track not associated with a charged particle)**
- Smaller radius of the innermost pixel layer → **closer to the interaction point** to improve tracking and vertexing performances
- Reduced material budget → **reduce multiple scattering**

Lorentz drift

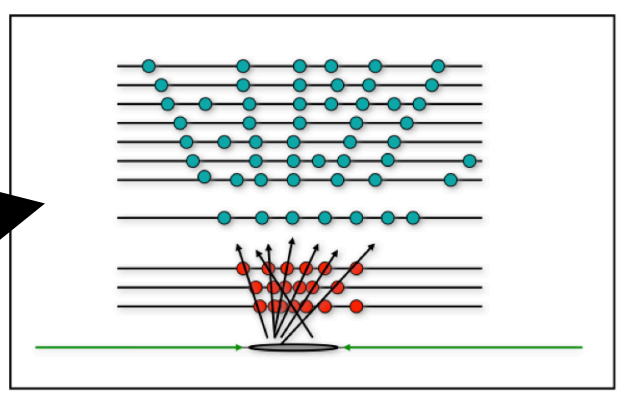
- **Deflection** of the drifting charge carriers
- Strongly depends on **irradiation (bulk damage and annealing)**, charge carrier mobility, etc.
- Operation under **specific temperature**
- **Alignment** procedure to correct for the effect

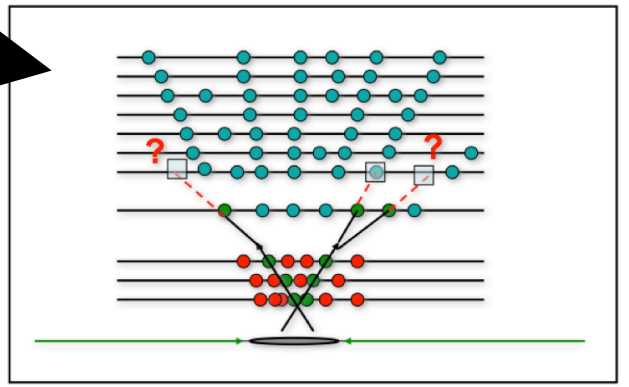


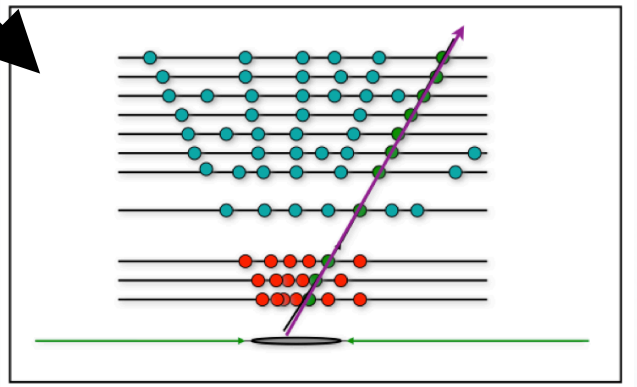
NIM A1037 (2022) 166795

Track reconstruction

- **Seeding**:
  start with **track candidates** and calculate initial trajectory parameters and their uncertainties
- **Building (pattern recognition)**:
  propagate track candidates to find new compatible hits and update track parameters
- **Fitting**:
  best estimate of track parameters for a smooth trajectory using combination of associated hits
- **Selection**:
  assign quality flags based on the χ^2 of the fit and the track compatibility with interaction region useful for rejection of fake tracks





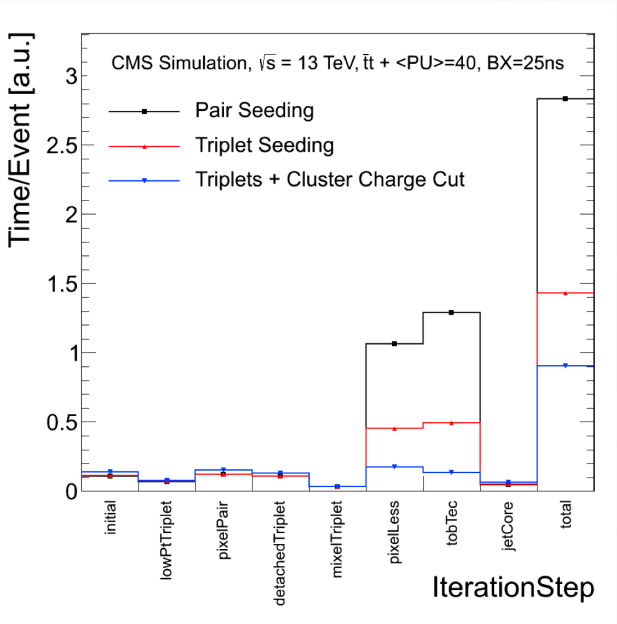


Seeding

arXiv:2304.05853

Iteration	Seeding	Target track
Initial	pixel quadruplets	prompt, high p_T
LowPtQuad	pixel quadruplets	prompt, low p_T
HighPtTriplet	pixel triplets	prompt, high p_T recovery
LowPtTriplet	pixel triplets	prompt, low p_T recovery
DetachedQuad	pixel quadruplets	displaced--
DetachedTriplet	pixel triplets	displaced-- recovery
MixedTriplet	pixel+strip triplets	displaced-
PixelLess	inner strip triplets	displaced+
TobTec	outer strip triplets	displaced++
JetCore	pixel pairs in jets	high-p_T jets
Muon inside-out	muon-tagged tracks	muon
Muon outside-in	standalone muon	muon

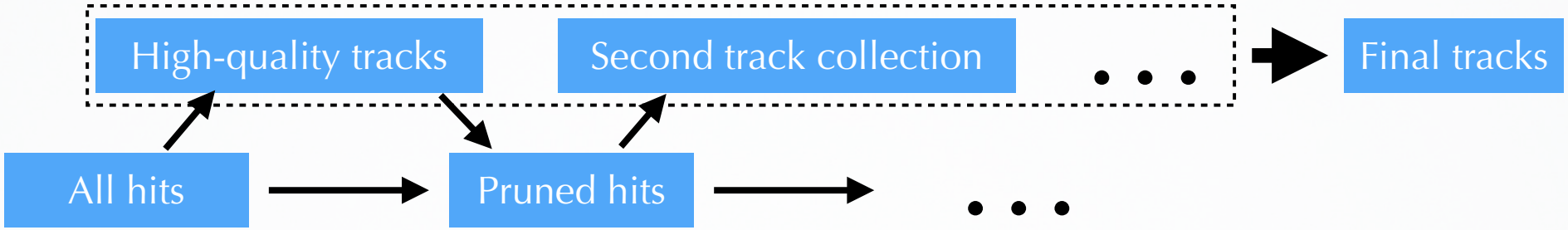
CMS-CR-2014-345



- Each interaction has its **specific seeding** setup
- Can reconstruct both **triplets** and **quadruplets** (and doublet for recovery)
- **Triplets** allow low-p_T tracks recovery but are more polluted by fakes
- For the outer Tracker seeds, the triplets are built via **doublet propagation** to a third compatible layer

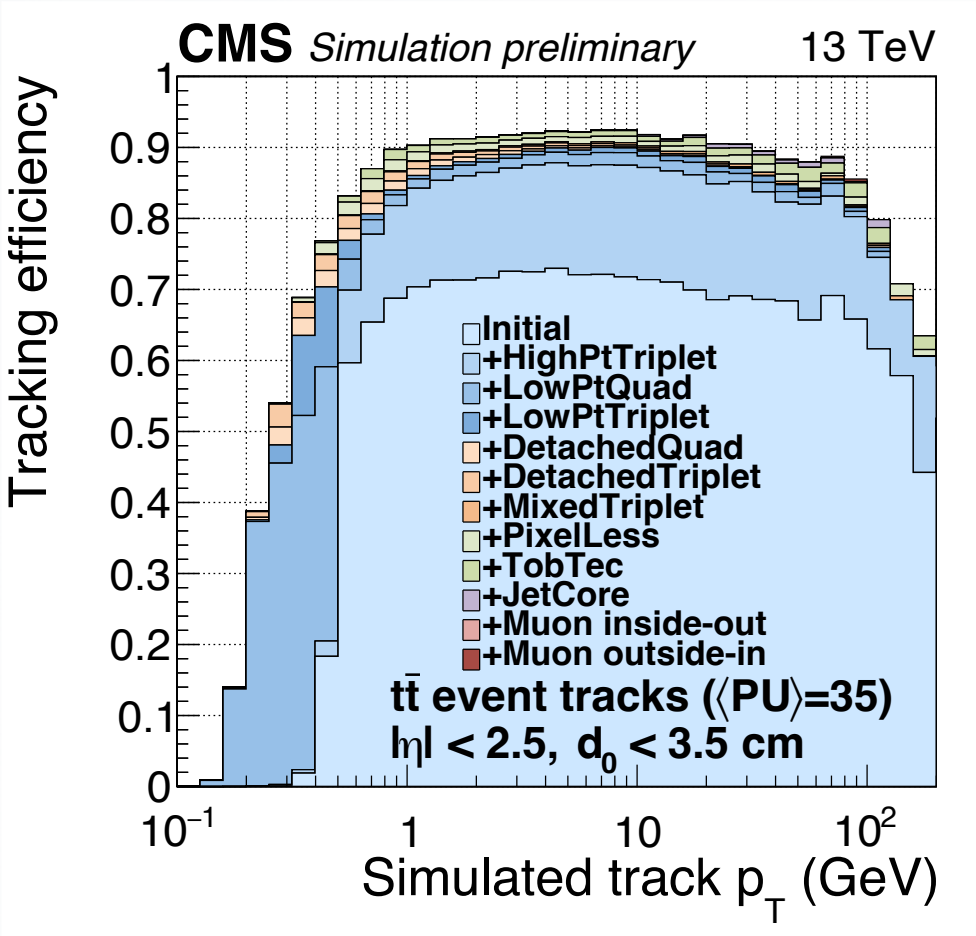
Iterative tracking

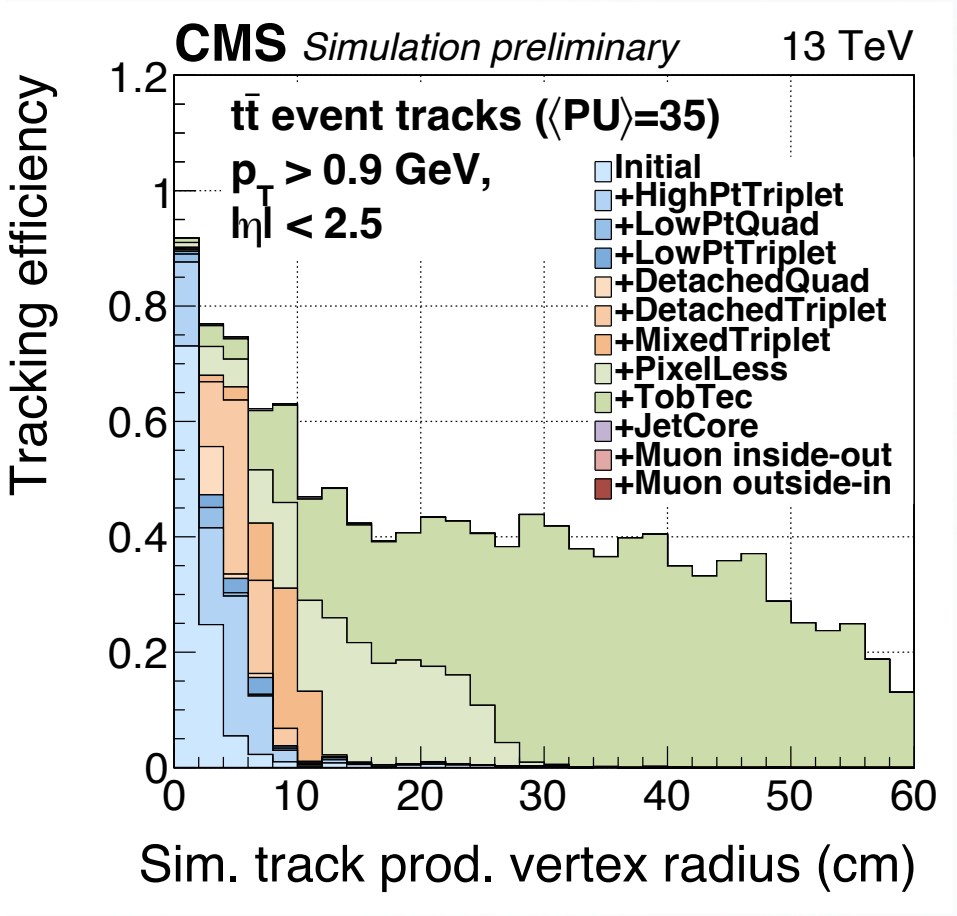
- Track reconstruction is an **iterative** procedure
  - ➡ Reconstruct **high-quality** tracks
  - ➡ **Remove** the hits associated with high-quality tracks from the hit collection
  - ➡ Use **remaining** hits to reconstruct other tracks



- Iterations
  - ➡ **Initial**: **high-p_T quadruplets** with high precision pixel hits compatible with the beamspot region
  - ➡ **Triplets**: recover acceptance at **low p_T** and **displacement**
  - ➡ **Strips**: use seeds from strips to find tracks **detached** from the primary vertex and those in **special phase space** regions

Iterative tracking





NIM A1037 (2022) 166795

Track selection

- Significantly reduce fake rate with a set of track quality requirements (Loose, Tight, **HighPurity**)
- Use the **reconstructed** variables:
  - ➡ Number of layers with hits
  - ➡ Goodness of track fitting (χ^2/N_{dof})
  - ➡ Compatibility with primary interaction vertex (including pileup vertices)
- Selected tracks from each of the iterations are **merged** into one collection
- The generalTrack collection contains **Loose** tracks
- The **HighPurity** tracks are typically used in physics analysis

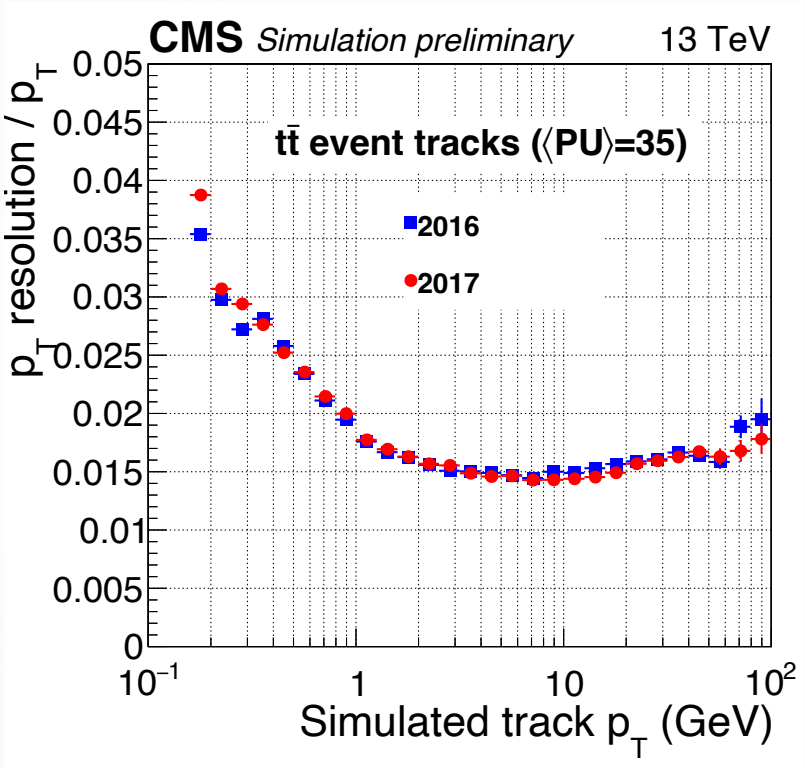
$$\chi^2/\mathrm{dof} < \alpha_0 N_{\mathrm{layers}}$$

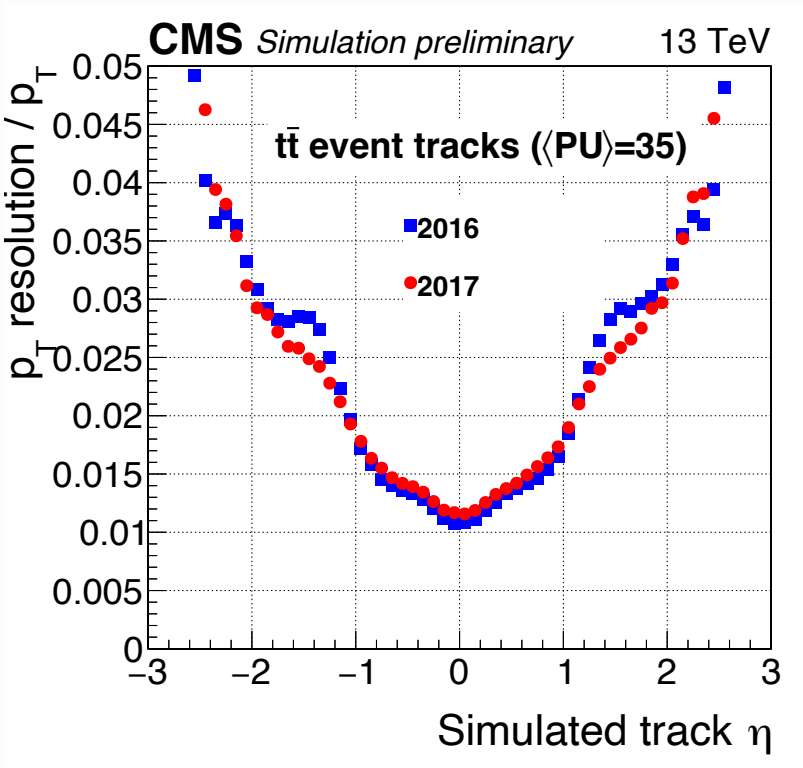
Iteration	Min layers			Min 3-D layers			Max lost layers			α_0		
	L	T	H	L	T	H	L	T	H	L	T	H
0 & 1	0	3	4	0	3	4	∞	2	2	2.0	0.9	0.9
2 Trk	4	5	5	0	3	3	∞	1	1	0.9	0.7	0.5
2 Vtx	3	3	3	0	3	3	∞	1	1	2.0	0.9	0.9
3 Trk	4	5	5	2	3	4	1	1	1	0.9	0.7	0.5
3 Vtx	3	3	3	2	3	3	1	1	1	2.0	0.9	0.9
4	5	5	6	3	3	3	1	0	0	0.6	0.4	0.3
5	6	6	6	2	2	2	1	0	0	0.6	0.35	0.25

JINST 9 (2014) P10009

Momentum resolution

- p_T resolution:
  - ➡ 3-4% at low p_T because of **multiple scattering**
  - ➡ Reaching 1.5% at high p_T for ≈ 10 GeV tracks
  - ➡ Degradation of resolution at high p_T due to **less bending** in magnetic field
  - ➡ Best resolution for central tracks

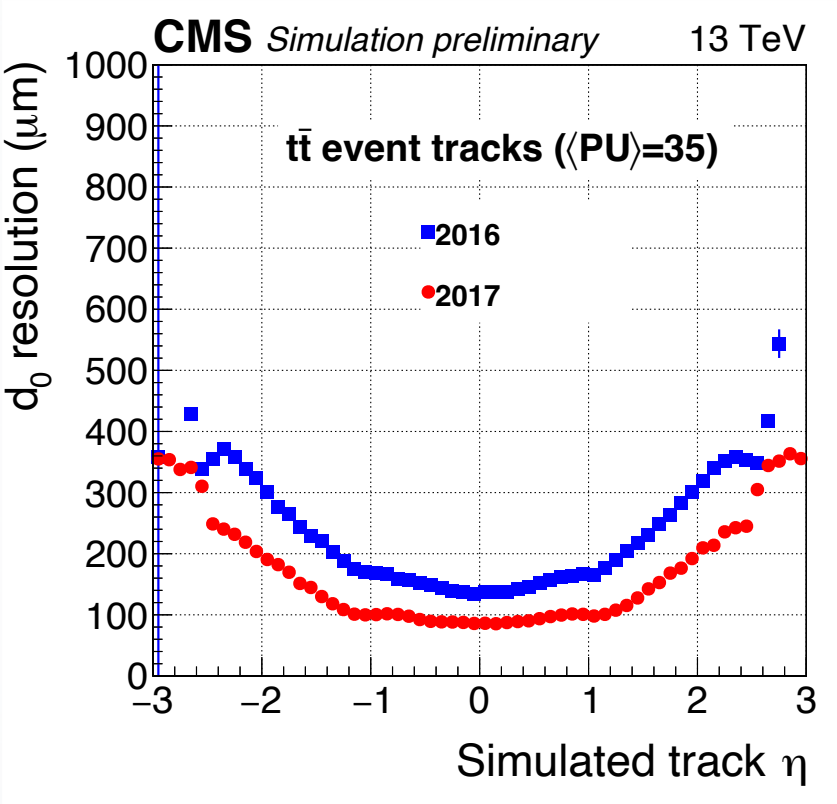


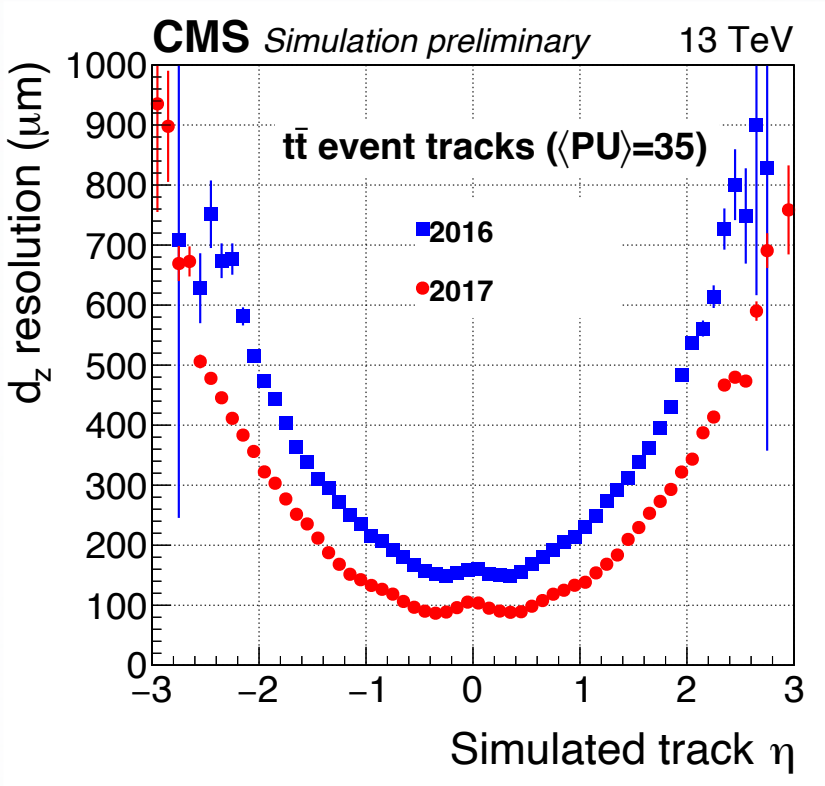


CMS-DP-2017-015

Impact parameter resolution

- Resolution reaches ≈ 100 μm for **central** tracks
- Degrades for **forward** tracks to up to ≈ 350 μm (> 500 μm) for **transverse** (**longitudinal**) impact parameters



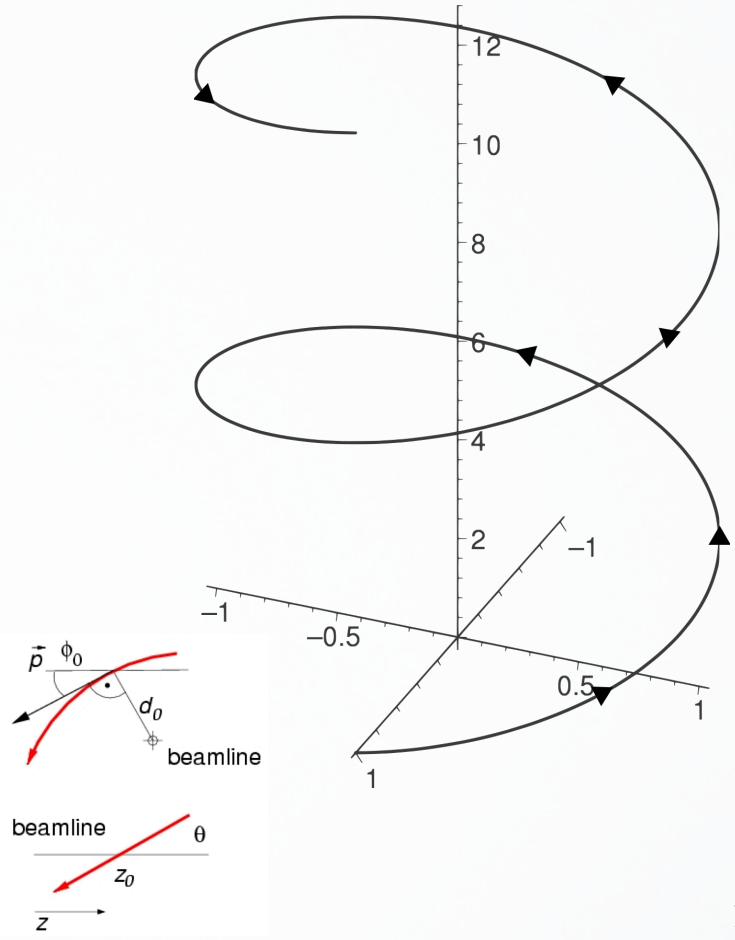


CMS-DP-2017-015

Trajectory parametrization

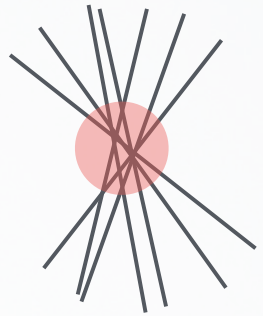
- A helical trajectory can be expressed by **five parameters**, but the parametrization is not unique
- Given one parametrization, we can **re-write** the same trajectory using another parametrization

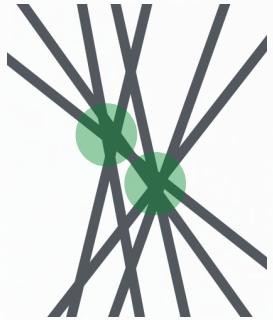
- **Signed radius** of curvature [cm]: proportional to particle's charge divided by the track's p_T [GeV]
- **Angle of trajectory** at a given point on the helix in the plane **transverse** to the beamline (φ)
- **Angle of trajectory** at a given point on the helix **along** the beamline (θ), also expressed as $\eta = -\ln(\tan(\theta/2))$
- **Impact parameter** relative to some reference point (e.g. beamspot or primary vertex) in the plane **transverse** to the beamline (d_{xy})
- **Impact parameter** in the plane **along** the beamline (d_z)



Vertexing

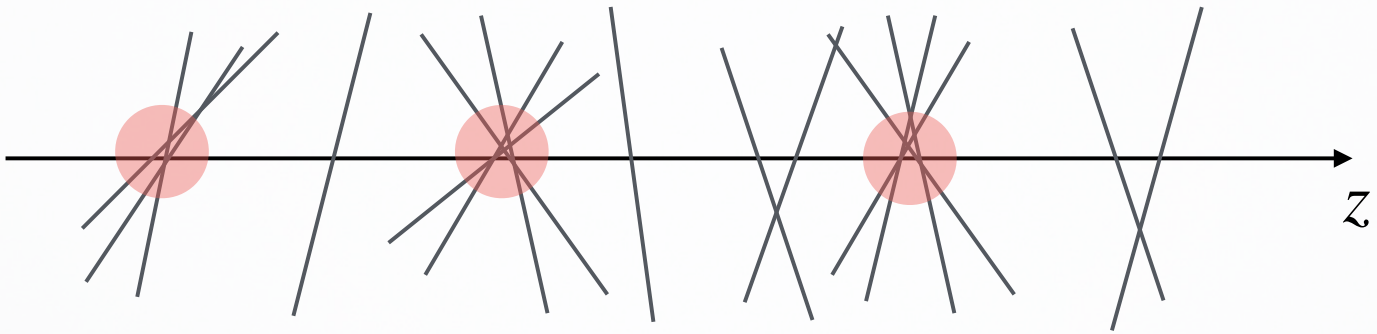
- Reconstruction and correct identification of the **vertex of the hard interaction** in event is of critical importance to correctly select final-state objects
- We also need to reconstruct **as many pileup vertices as possible** to allow an efficient pileup suppression with PUPPI or Charge Hadron Subtraction (CHS) algorithms
- The vertexing algorithm selects **good tracks originating from the interaction region** around the beamspot and **clusters** them according to the z coordinate of their point of closest approach (PCA) to the center of the beamspot
- When we cluster tracks into vertices, at the same time we want to resolve nearby vertices to separate the primary interaction from pileup vertices and **avoid vertex merging**
- At the same time, we **should avoid splitting** a genuine vertex in two!





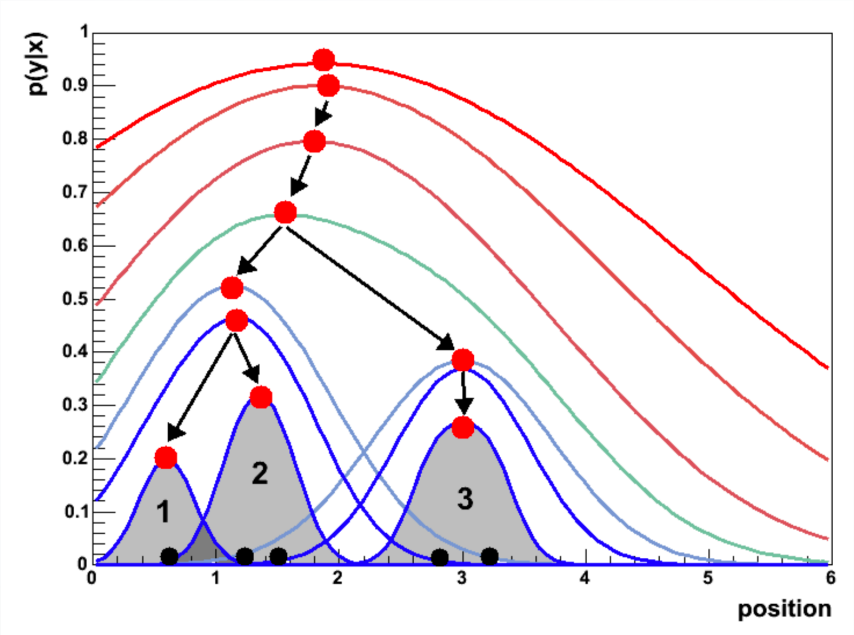
Gap clustering algorithms

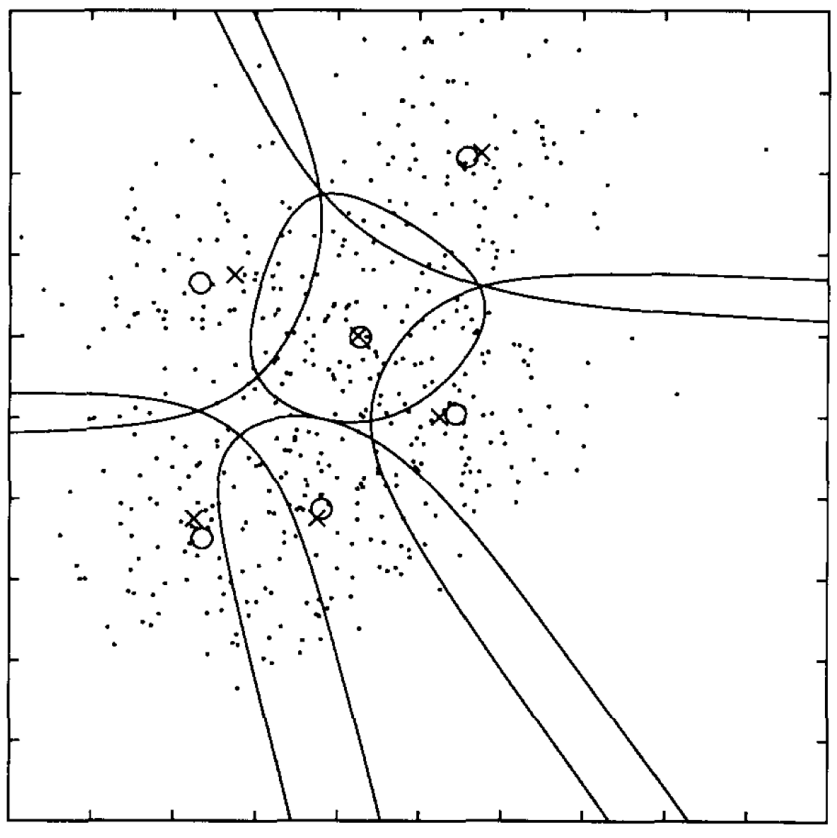
- A large number of algorithms are available to **cluster tracks into vertices** (K-means, Deterministic Annealing, etc.)
- Let us consider a simple gap clustering algorithm used at **HLT** as DivisiveVertexFinder:
  - ➡ Start by clustering tracks, which are sorted in z position
  - ➡ Consider all tracks to be the part of the same vertex
  - ➡ When any two neighboring tracks are having a gap exceeding a given threshold (e.g. 5 mm), the vertex is split
- The algorithm is **simple** and **fast** but **not optimal at high pileup**



Deterministic Annealing

- **Offline vertex clustering** uses Deterministic Annealing algorithm
- Inspired by **thermodynamics**: find the **global minimum** with many degrees of freedom analogous to a physical system approaching a state of **minimal energy** through a series of gradual temperature reductions



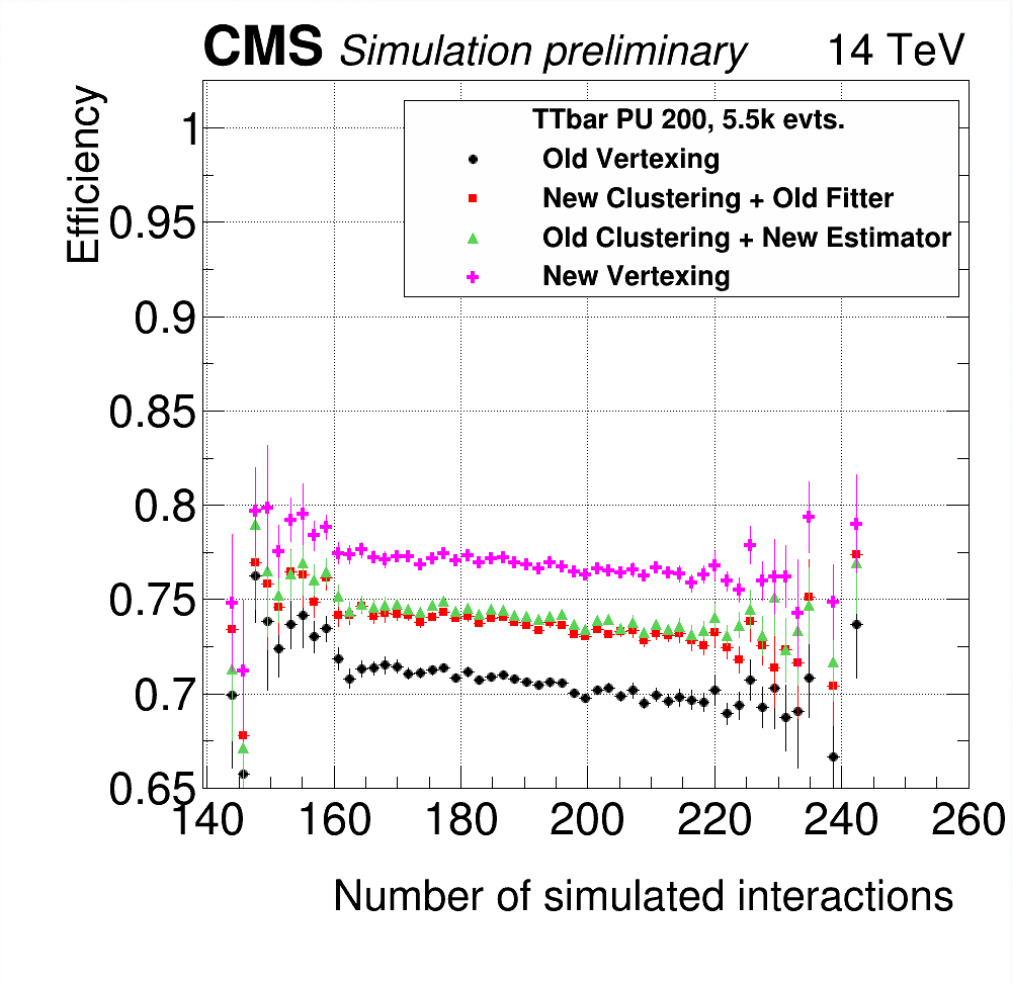


Patt. Rec. Lett. 11 (1990) 589

- All tracks from one vertex (**high** temperature)
- Split vertices below a (**low**) critical temperature
- **Iterative procedure** to balance between vertex **merging** and **splitting**

Deterministic Annealing

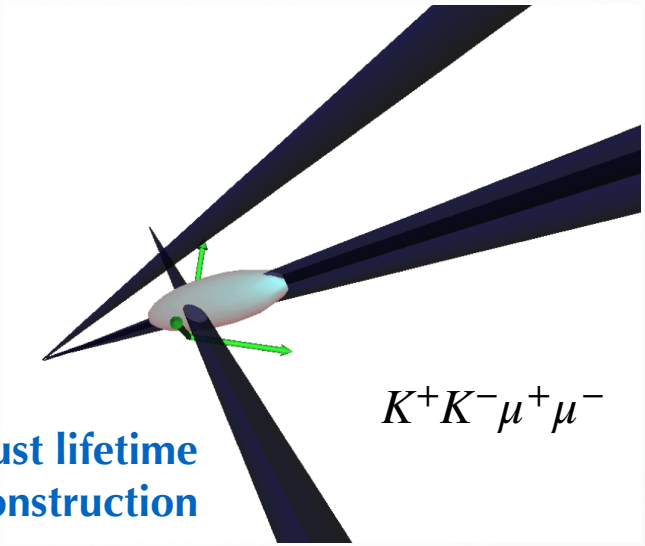
- Recently improved for **HL-LHC** by **clustering tracks in blocks**, also ported to heterogenous architectures
- Apply DA to the blocks with **512 tracks** each, clustered in z
- **Better** efficiency and timing performance



CMS-DP-2022-052

Vertex fitting

- With the tracks clustered in vertices, the **3D position of the vertex** can be fitted
- A good candidate for a vertex fitter is a **Kalman Filter** - the least-squares estimator, which minimizes the sum of the squared standardized distances of all tracks from the vertex position
- Can be used **iteratively**, refitting the tracks with taking into account the vertex position
- Does not properly handle **outlier tracks**, leading to bad fits if tracks are included in the vertex to which they do not belong, or rejected in the opposite case
- A better approach: **Adaptive Vertex Fitter (AVF)** used in CMS offline reconstruction
  - ➡ Each track is assigned a **weight** representing the probability that it belongs to the vertex
  - ➡ This allows to **down-weigh the outlier tracks**, making the vertex fit much more robust



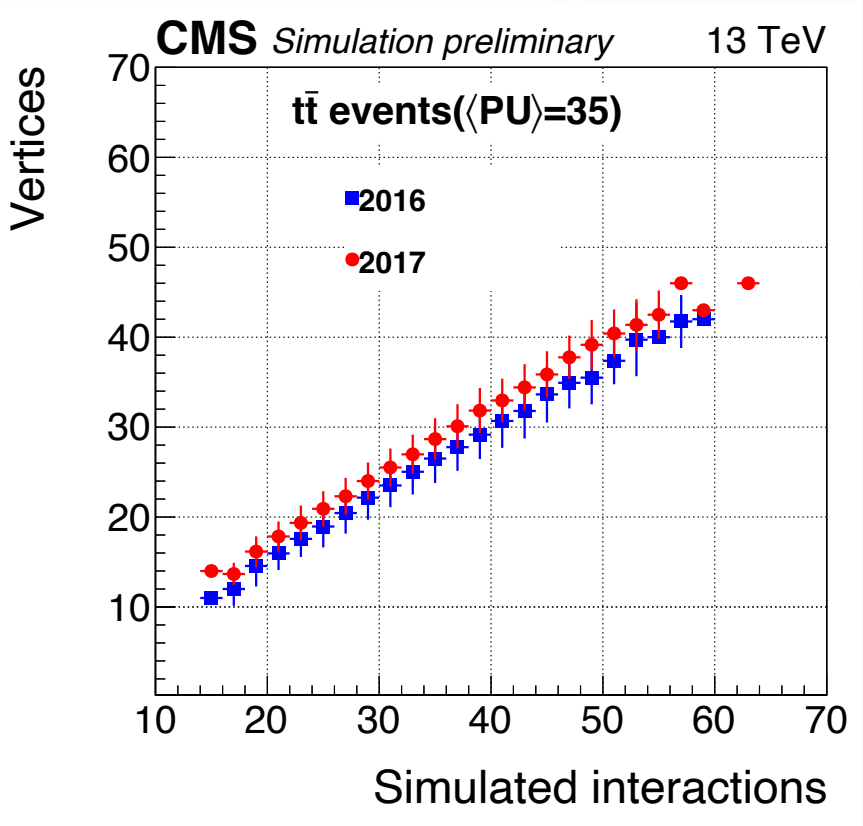
robust lifetime reconstruction

CMS-NOTE-2007-008

Vertexing performance

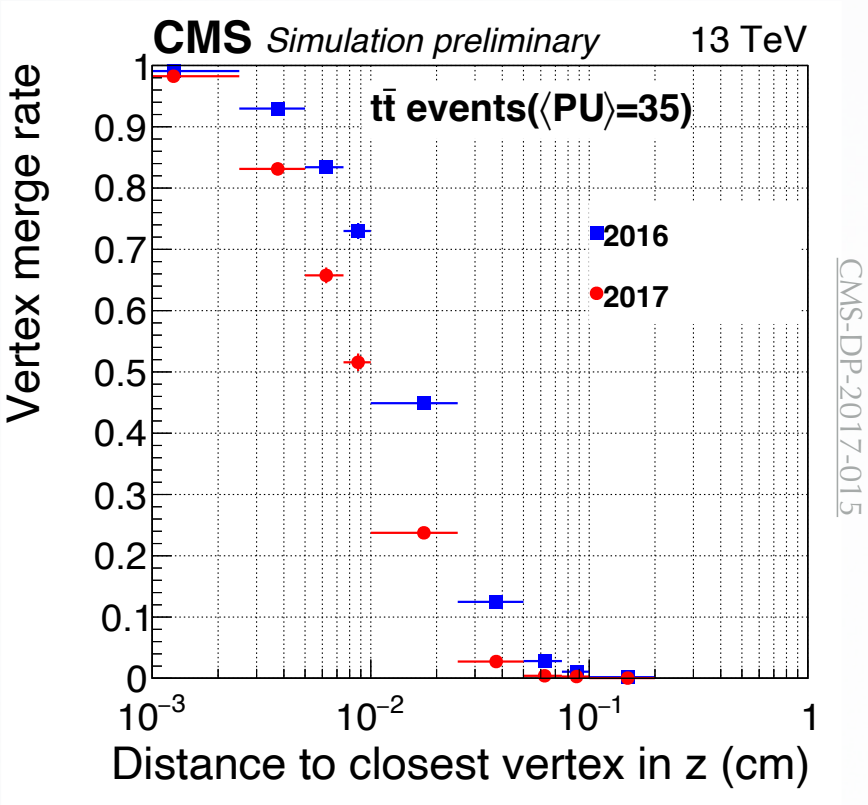
- Vertex reconstruction efficiency increased significantly after the **Phase-I pixel detector upgrade**
- ≈ 85% at low pileup, **decreasing with the number of interactions** (when computed for all pileup vertices)
- Much higher efficiency for the **hard-interaction vertex**
- A **linear relation** between the number of reconstructed vertices and the number of pileup interactions
- The efficiency loss at high pileup attributed to the **vertex-merging** effects

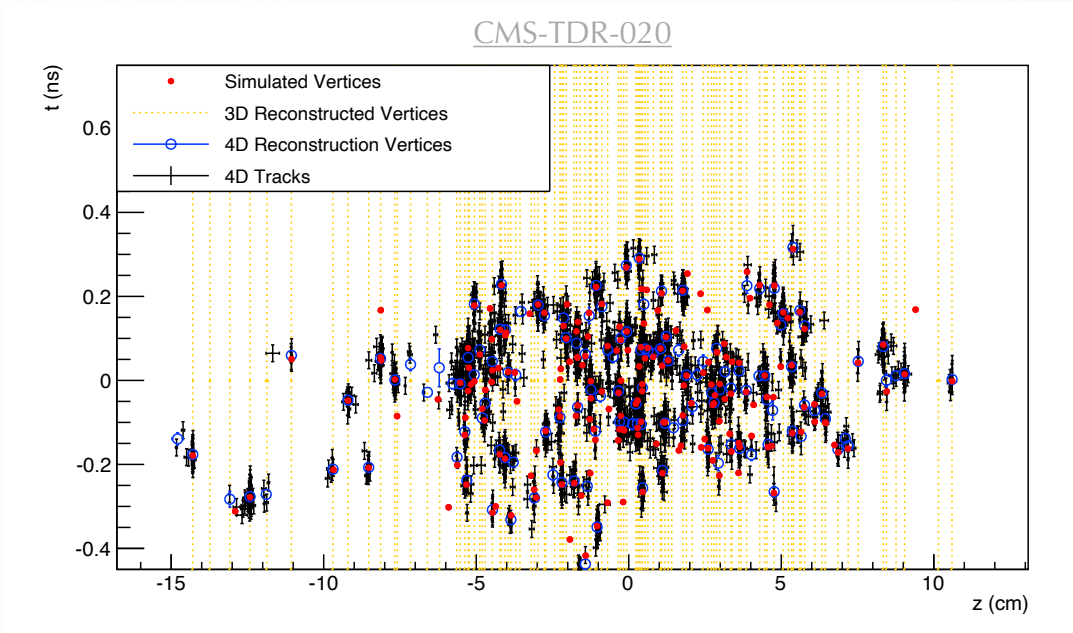
CMS-DP-2017-015



Vertexing performance

- **Merging** of vertices starts for distances closer than **0.3 mm**
- Vertices closer than **0.1 mm** will be **merged** into a single reconstructed vertex
- Future: CMS Phase-2 timing layer will help resolve overlapping vertices





Tracks in CMS data formats

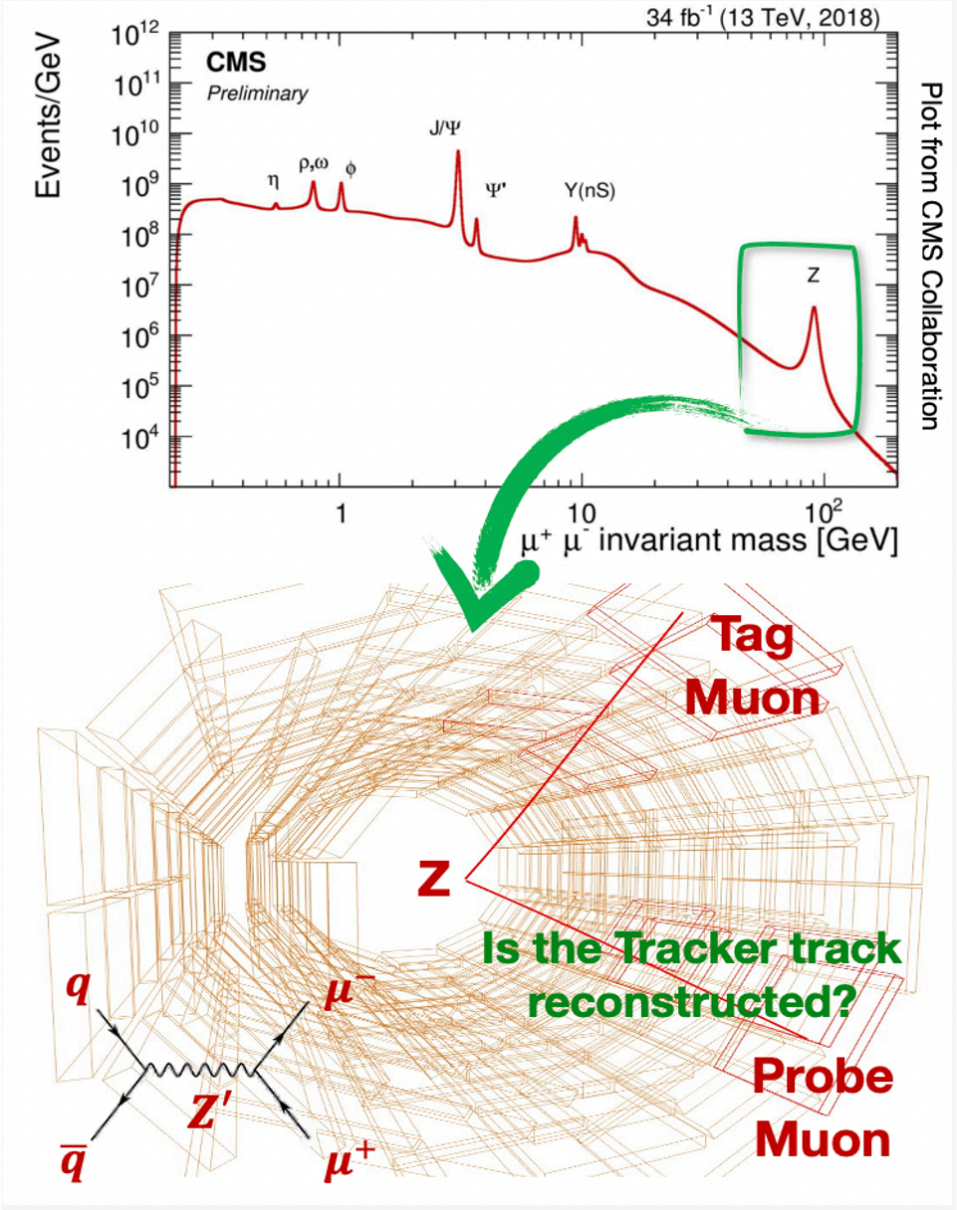
- CMS provides several **different data formats** for physics analysis
- In general, we want to **reduce as much information as possible** to save storage space and speed up event processing
- However, this means significant **compromises** in the accessibility of « low-level » information, such as individual tracks
- **Data formats**:
  - ➡ **AOD**: Save all reconstructed objects and drop only low-level detector information. All tracks passing loose selection requirements are saved in the **generalTracks** collection. Note that the use of this format is to be avoided if at all possible as it can be tricky to access it with only a few copies available for analysis access.
  - ➡ **MiniAOD**: Stripped down version of AOD. Keeps only high-level objects necessary for most analyses. No single collection of all tracks is kept. Tracks associated with PF candidates are available through **packedPFCandidate** object. Tracks not associated with PF candidates are stored in the **lostTracks** collection if they have $p_T > 0.95$ GeV or if they are associated with a secondary vertex or a K or Λ candidate.
  - ➡ **NanoAOD**: Store only the most relevant information for analysis as **flat tuples**. Track information is stored only for some **isolated tracks**. NanoAOD content documentation

Vertices in CMS data formats

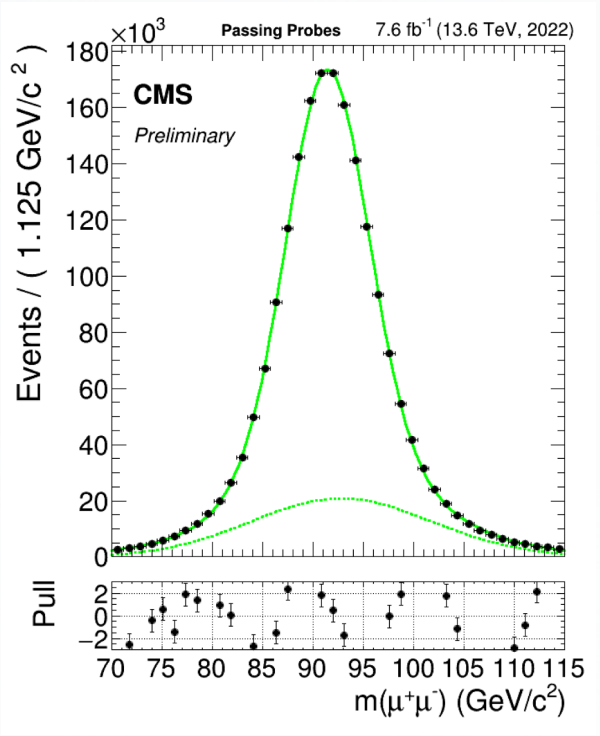
- **Similar strategy** as used for saving track information.
- **Data formats**:
  - ➡ **AOD**: The **offlinePrimaryVertices** collection contains all reconstructed vertices.
  - ➡ **MiniAOD**: To save space, the **offlineSlimmedPrimaryVertices** drop the references from the vertices to the associated tracks and the numerical precision of vertex parameters is reduced.
  - ➡ **NanoAOD**: Basic information about the **main primary vertex** is available, as well as the number of additional PVs. Some information about secondary vertices is also available. NanoAOD content documentation

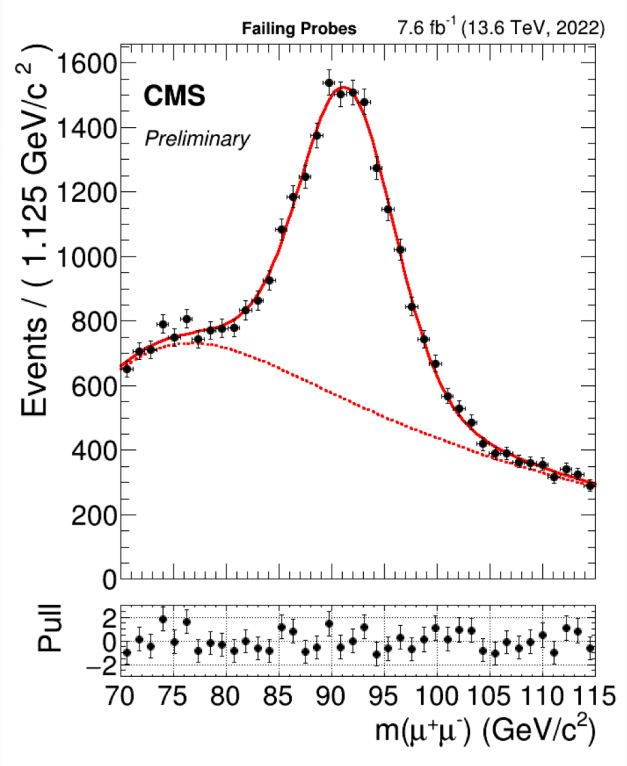
Tag & Probe method

- Identify **similar** objects in **data** and **MC** to validate MC predictions
- Use **known resonances** decaying to two muons, produced copiously and measured with high precision
- Define a **tag** muon reconstructed by both Tracker and the Muon system with tight requirements
- Define a **probe** muon reconstructed by the Muon system passing loose selection
- Select a **good dimuon candidate** with additional kinematic requirements
- Check if a probe muon can be **matched** to at least one **track in the Tracker** in some cone around the direction of the muon
- Calculate the **track reconstruction efficiency**



Tag & Probe method (cont.)





efficiency = (99.32 ± 0.06) %

--- parameters

Passing Probes

- $\alpha P = (13945.05 \pm 7.12) \times 10^{-2}$
- $\beta P = (8.00 \pm 0.00) \times 10^{-2}$
- $\gamma P = (61.74 \pm 0.16) \times 10^{-2}$
- $\mu P = (9138.22 \pm 0.62) \times 10^{-2}$
- $nBkgP = (412.36 \pm 11.32) \times 10^{3}$
- $nSigP = (1533.45 \pm 9.82) \times 10^{3}$
- $\sigma P = (3704.67 \pm 7.08) \times 10^{-3}$
- $\Gamma P = (228.02 \pm 3.70) \times 10^{-2}$

Failing Probes

- $\alpha F = (6688.51 \pm 126.60) \times 10^{-2}$
- $\beta F = (8.00 \pm 0.81) \times 10^{-2}$
- $\gamma F = (2.83 \pm 0.15) \times 10^{-2}$
- $\mu F = (9139.77 \pm 10.86) \times 10^{-2}$
- $nBkgF = (21.36 \pm 0.98) \times 10^{3}$
- $nSigF = (10.49 \pm 0.98) \times 10^{3}$
- $\sigma F = (3991.92 \pm 489.43) \times 10^{-3}$
- $\Gamma F = (253.40 \pm 186.70) \times 10^{-2}$

CMS-DP-2022-046

- **Background** events with muons coming not from the resonance can **bias** the measurement
  - ➡ Perform a simultaneous fit using signal and background templates to data
  - ➡ Subtract the fitted number of background events from data observation
  - ➡ Measure efficiency in data
  - ➡ Calculate the ratio between the measured efficiencies for signal muons in data and MC

Additional information

- **Technical information**
  - ➡ Tracking performance in Run 1, JINST 9 (2014) P10009
  - ➡ Track reconstruction with mkFit in Run3, CMS-DP-2022-018
  - ➡ SW guide documentation: TrackReco, VertexReco
  - ➡ Tracking Training Day, agenda (2019)
  - ➡ Tracker Training Days, agenda (2023)
  - ➡ Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors, R. Frühwirth and A. Strandlie

- **Contact information**
  - ➡ Tracking POG TWiki
  - ➡ CMS Talk forum

The exercise

- **Learn how to:**
  - ➡ Extract basic track parameters
  - ➡ Select tracks for analysis using track quality cuts
  - ➡ Reconstruct invariant masses from tracks
  - ➡ Extract basic parameters for primary vertices
  - ➡ Reconstruct secondary vertices and composite particles
  - ➡ Compute track reconstruction efficiency with Tag & Probe method

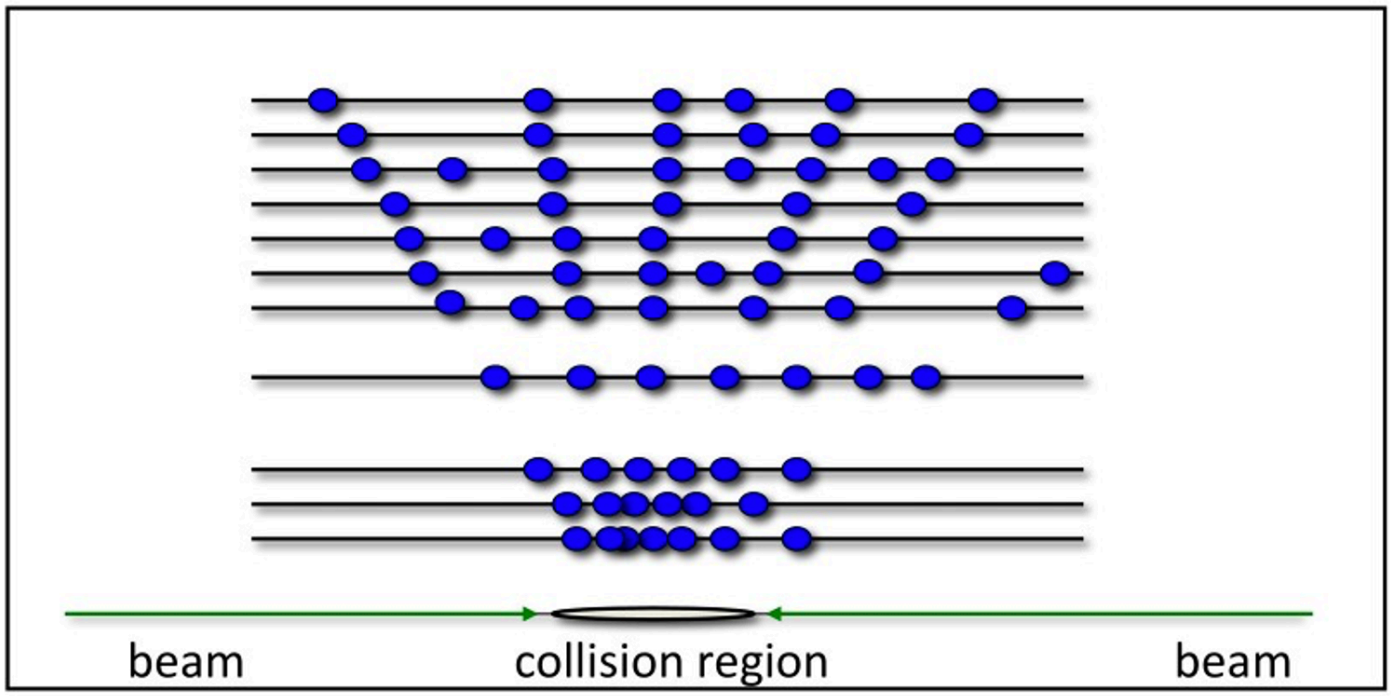
- **Useful links:**
  - ➡ Instructions for the exercise

BACKUP

Track reconstruction: overview

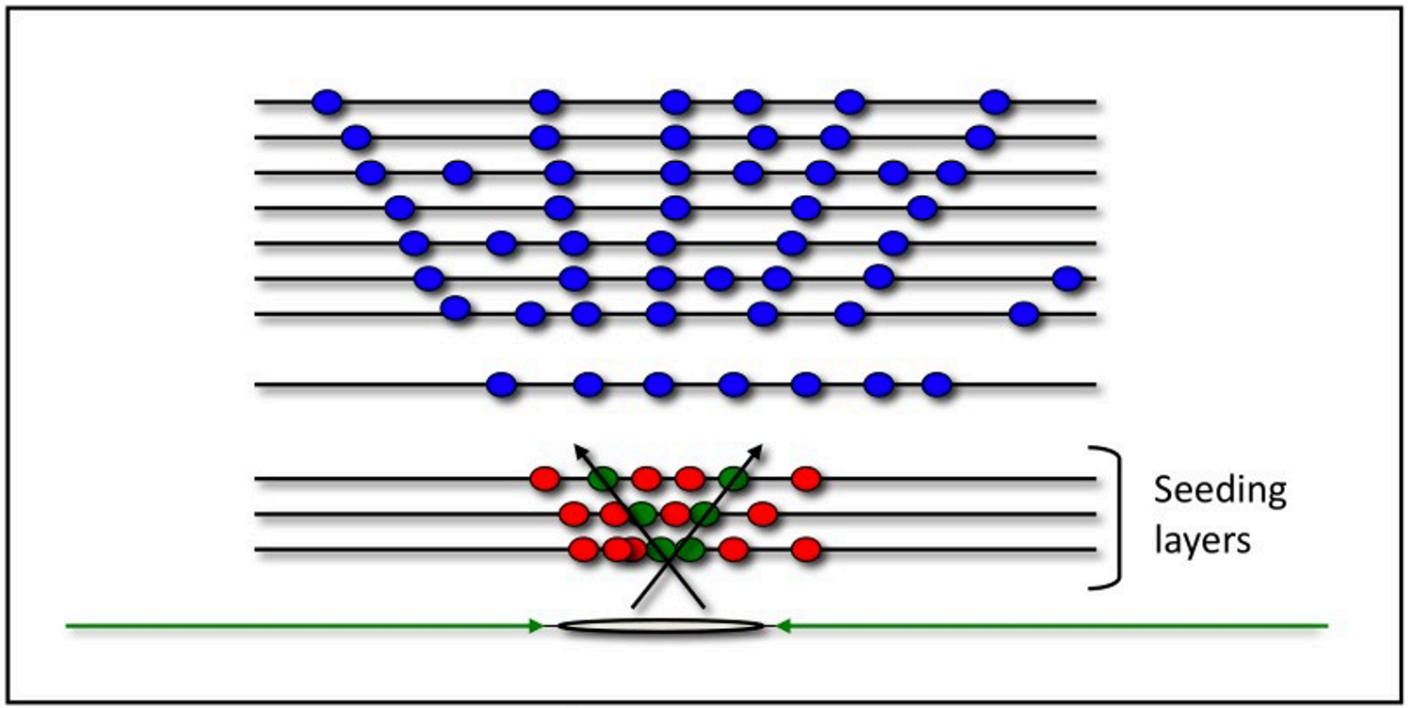
Local Tracker reconstruction

- Pixel and strip signals are clustered into « **hits** »
- Determine a « **coarse** » position and corresponding error matrix of each hit



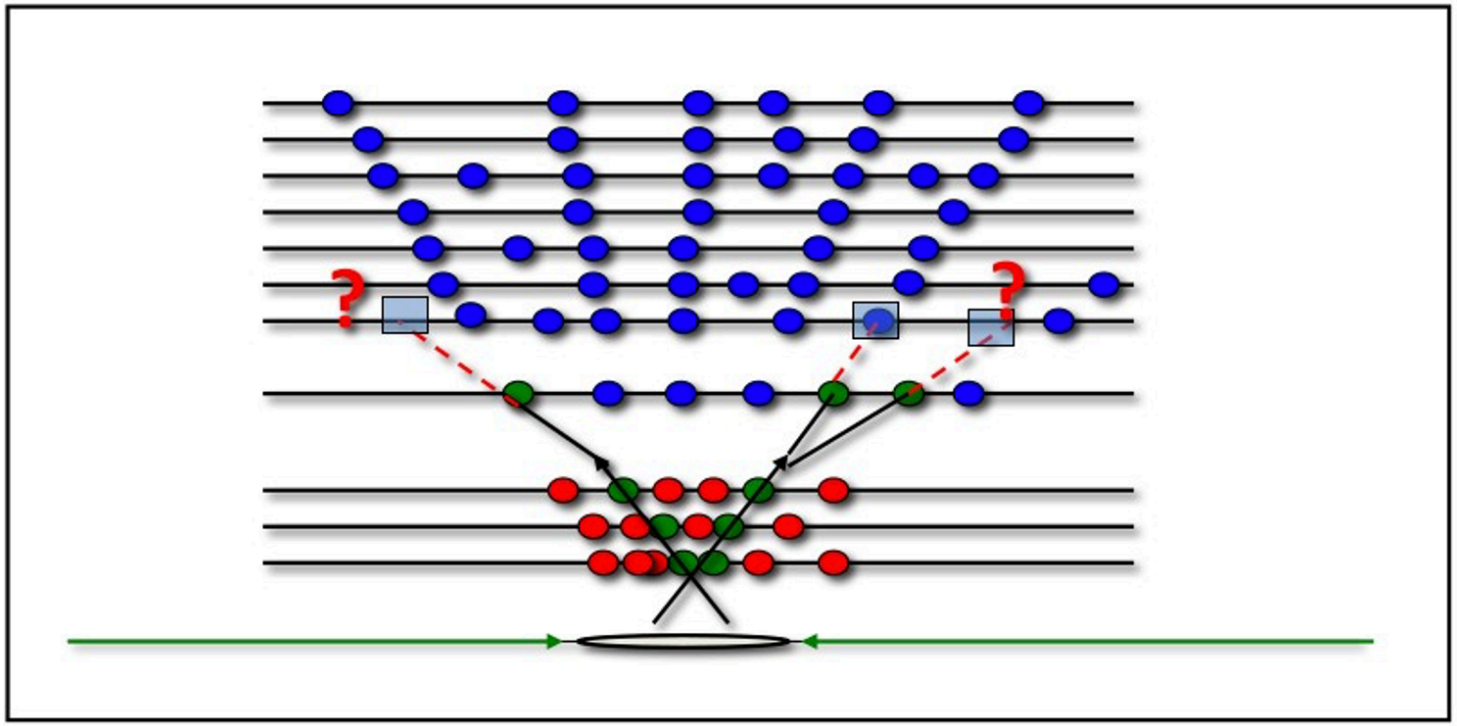
Trajectory **seeding**

- **Initial** estimate of trajectory parameters from a small subset of measurements, i.e. the hits on the **seeding** layers of the detector
- Build seeds in **external** layers in next iterations
- **TrajectorySeed**



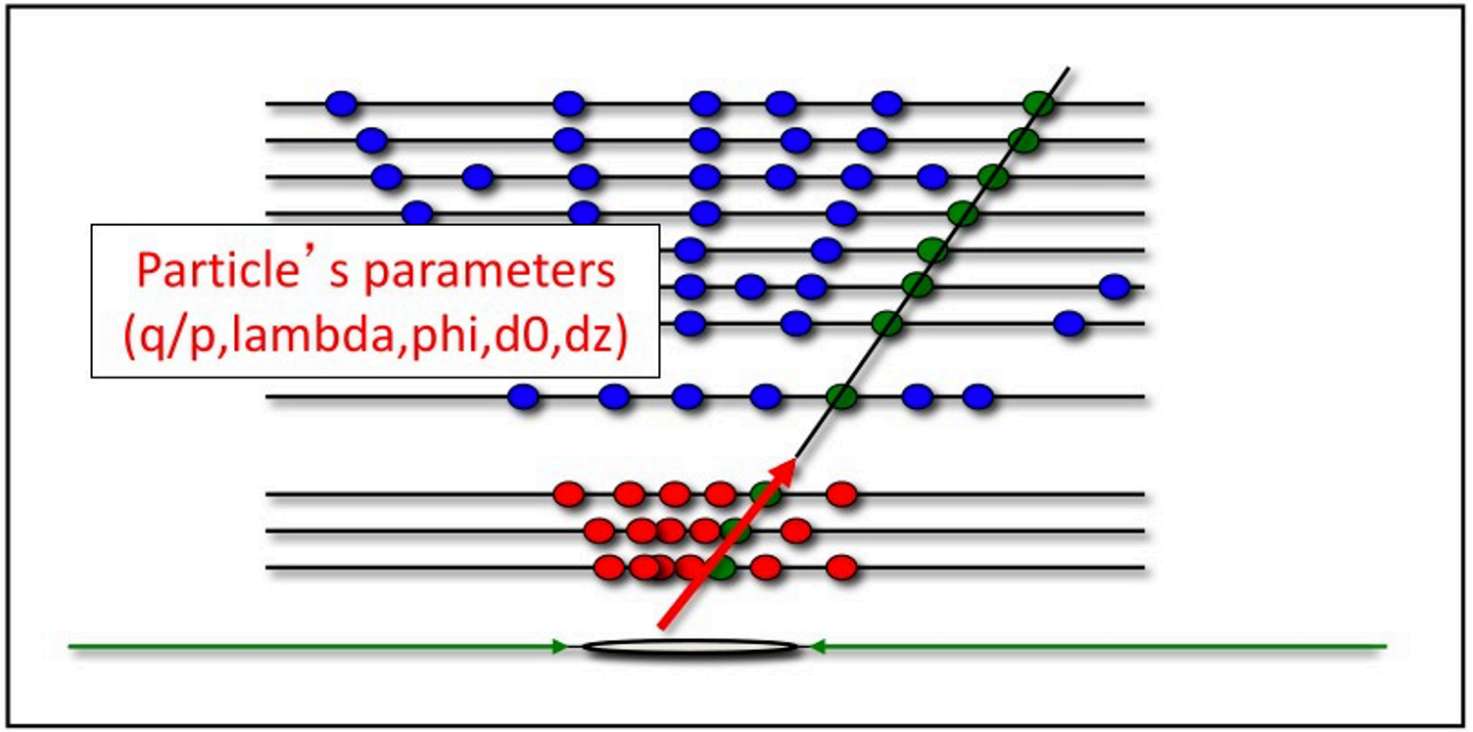
Trajectory **building**

- **Iteratively** collect all hits originating from the same charged particle
- **TrackCandidate**



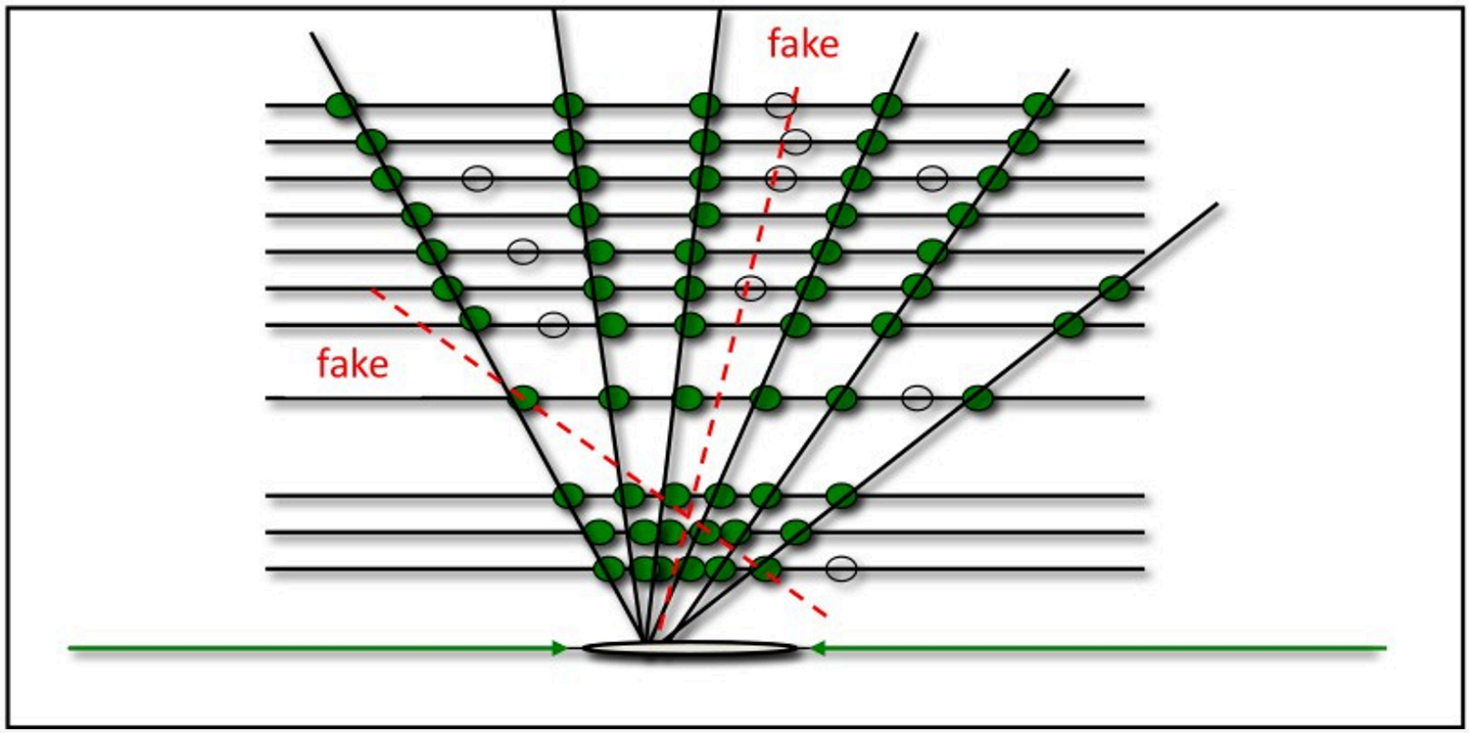
Trajectory **fitting**

- Estimation of the **final track parameters** from the KalmanFilter+Smoother fit using the full set of hits associated to the same charged particle
- **reco::Track**



Track **filtering**

- ➡ Remove **fake** or **badly reconstructed** tracks by using a BDT-based selection trained for each iteration
- ➡ **reco::Track (reduced)**



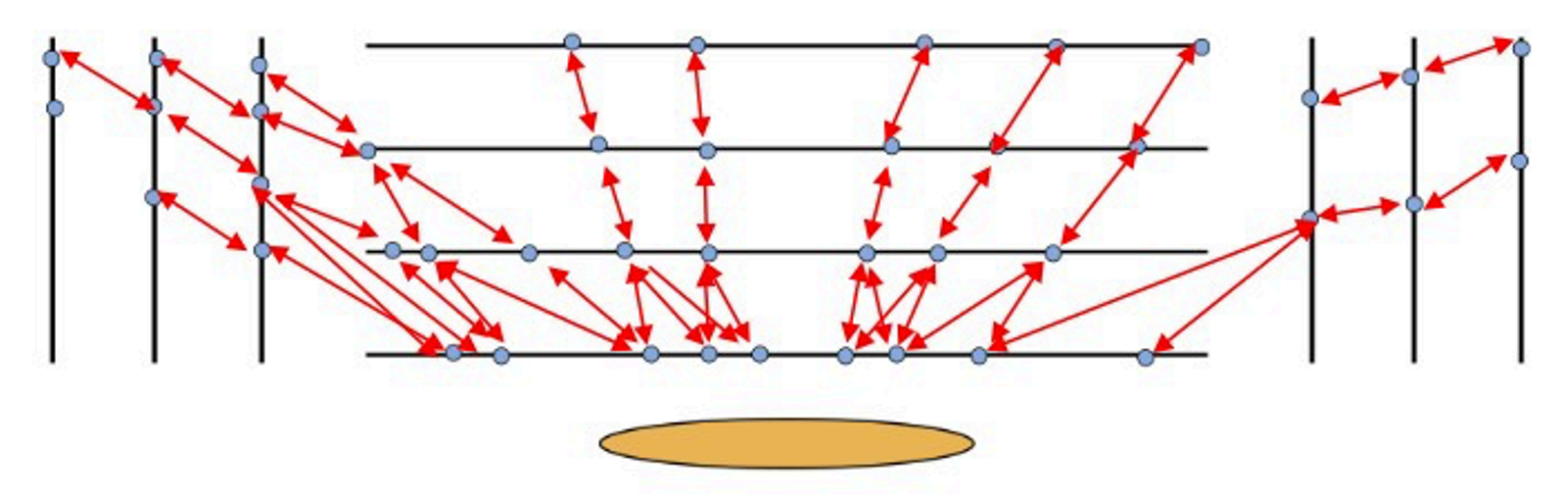
Seeding

Derivation of seeds

- **Pattern recognition** needs an **initial estimate** of the track parameters as a starting point
- Can be derived from a **small number of hits** in a subset of detectors
- The **choice of layers** to look for seeds is the **main difference** between tracking iterations
- **Best initial seeds** can be derived from **pixel** hits due to their excellent spatial resolution
- Hits from the **outer Tracker** and the **Muon system** can also be included

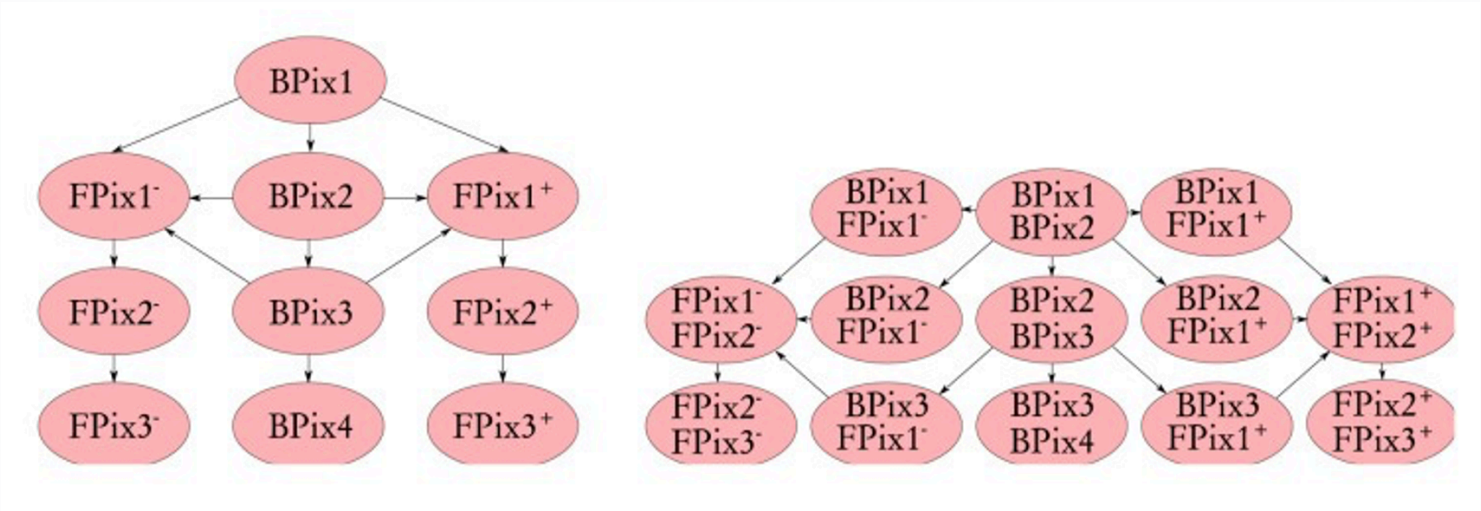
Cellular Automaton

- **Cellular Automaton (CA)** is a tracking algorithm designed for parallel architectures
- Requires a list of **layers** and their **pairings**:
  - ➡ A **graph** of all possible connections between layers is created
  - ➡ **Doublets** (or Cells) are created for each pair of layers, compatible with a region hypothesis
  - ➡ **Fast computation** of the compatibility between two connected cells
  - ➡ No knowledge of the world outside adjacent neighboring cells required - **easy to parallelize**



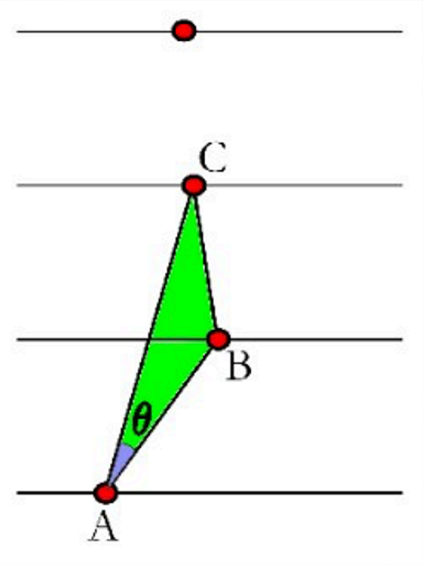
Cells

- Build **interconnections** among seeding layers
- Hit doublets for each layer pair can be computed **independently** in separate threads



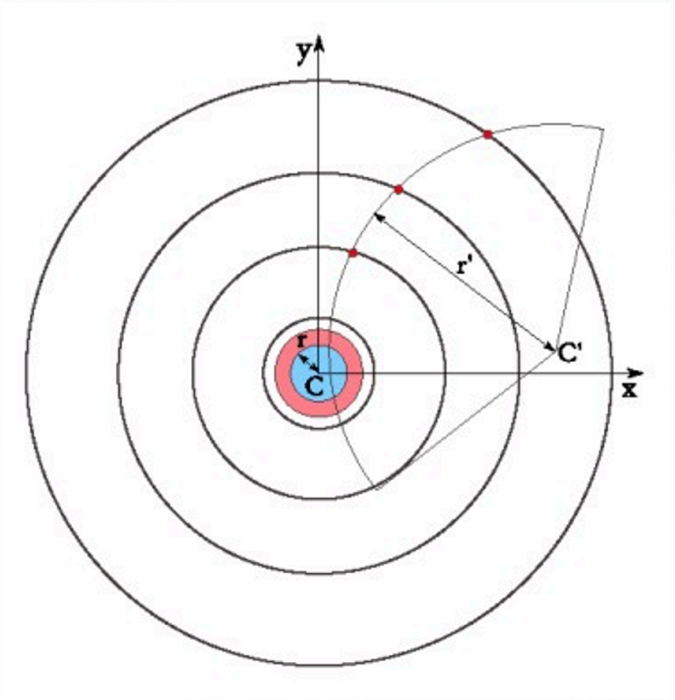
R-z plane compatibility

- The **compatibility** between two cells is checked only if they share one hit
  - ➡ AB and BC **share** hit B
- In the R-z plane require the **alignment** of the two cells
  - ➡ There is **maximum** value of θ that depends on the minimum value of the **momentum range** that we would like to explore

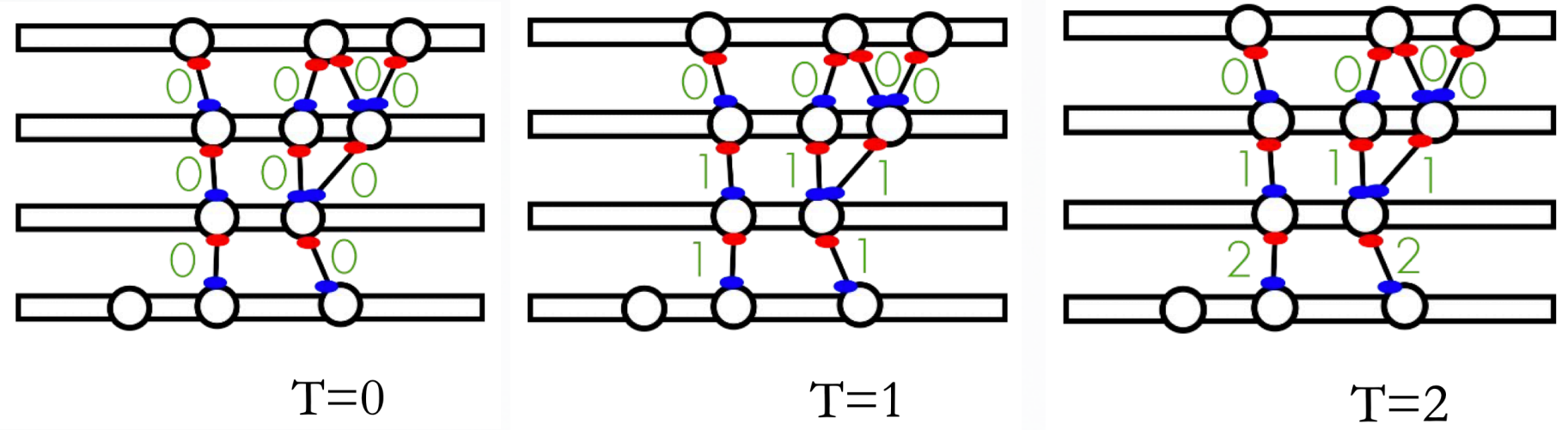


x-y plane compatibility

- In the **transverse plane**, the intersection between the circle passing through the hits forming the two cells and the beamspot is checked
  - ➡ They intersect if the distance between the centers d(C, C′) satisfies: $r'-r < d(C, C') < r'+r$
  - ➡ Since it is an out-in propagation, a tolerance is added to the beamspot radius (in red)
- One could also ask for a **minimum value** of transverse momentum and reject low values of r′



Evolution



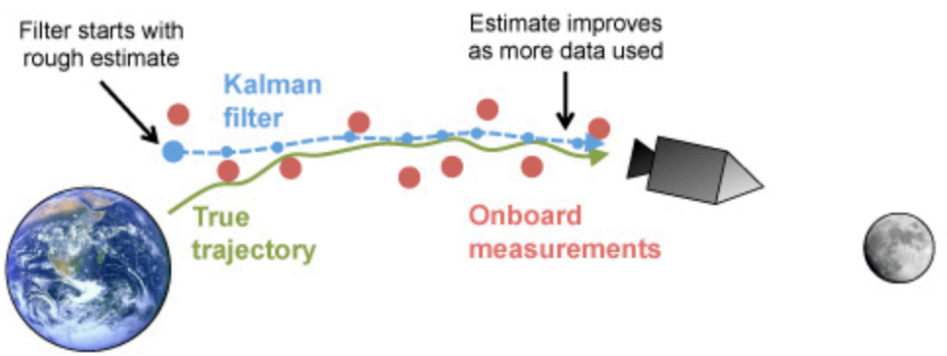
- If two cells satisfy all compatibility requirements, they become **neighbors** with a **state 0**
- In the **evolution** stage, their **state increases** in discrete generations if there is an outer neighbor with the same state
- At the end of the evolution stage, the state of the cells will contain the information about the **length**
- If one is interested in quadruplets, there will be one starting from a state 2 cell, pentuplets state 3, etc.

Trajectory building & fitting

Kalman Filter

Kalman Filter (1958)

Almost all modern control systems—both military and commercial—use the Kalman filter. It guided the Apollo 11 lunar module to the moon's surface and will guide the next generation of aircraft as well.



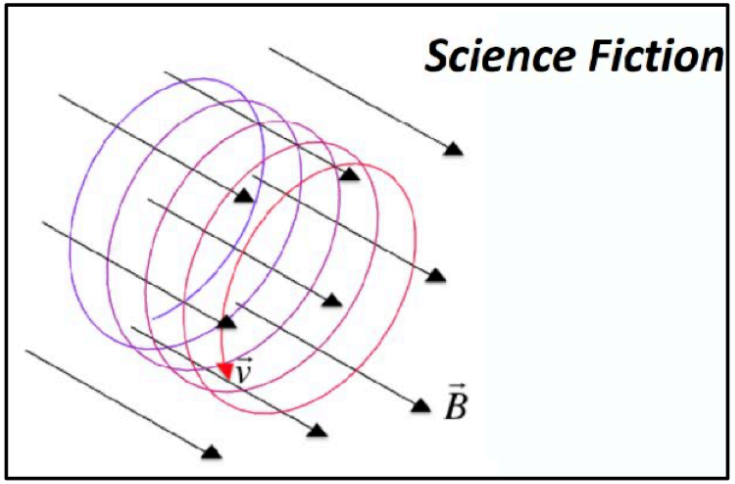
The Kalman filter compares predictions using Newton's Laws with onboard measurements to generate a better estimate of the spacecraft's true position.

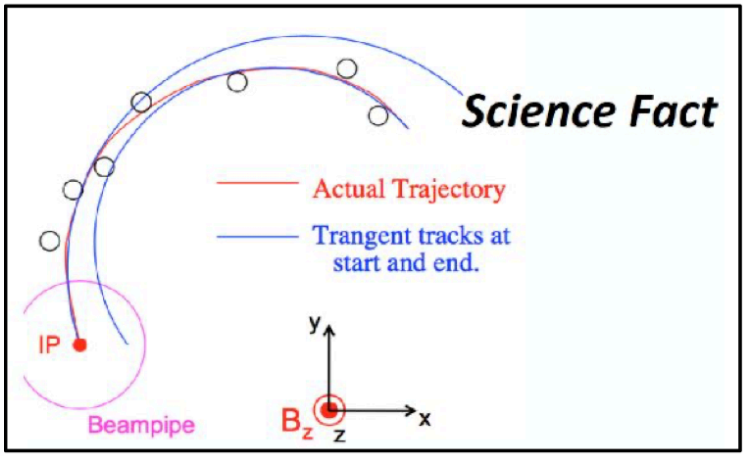
Kalman Filter in HEP

- **Expectation**: particle's trajectory can be described by a single helix
- **Reality**:
  - the B-field is not uniform
  - the processes of scattering and energy loss introduce additional stochastic effects

  ➡ Trajectory is a helix only locally

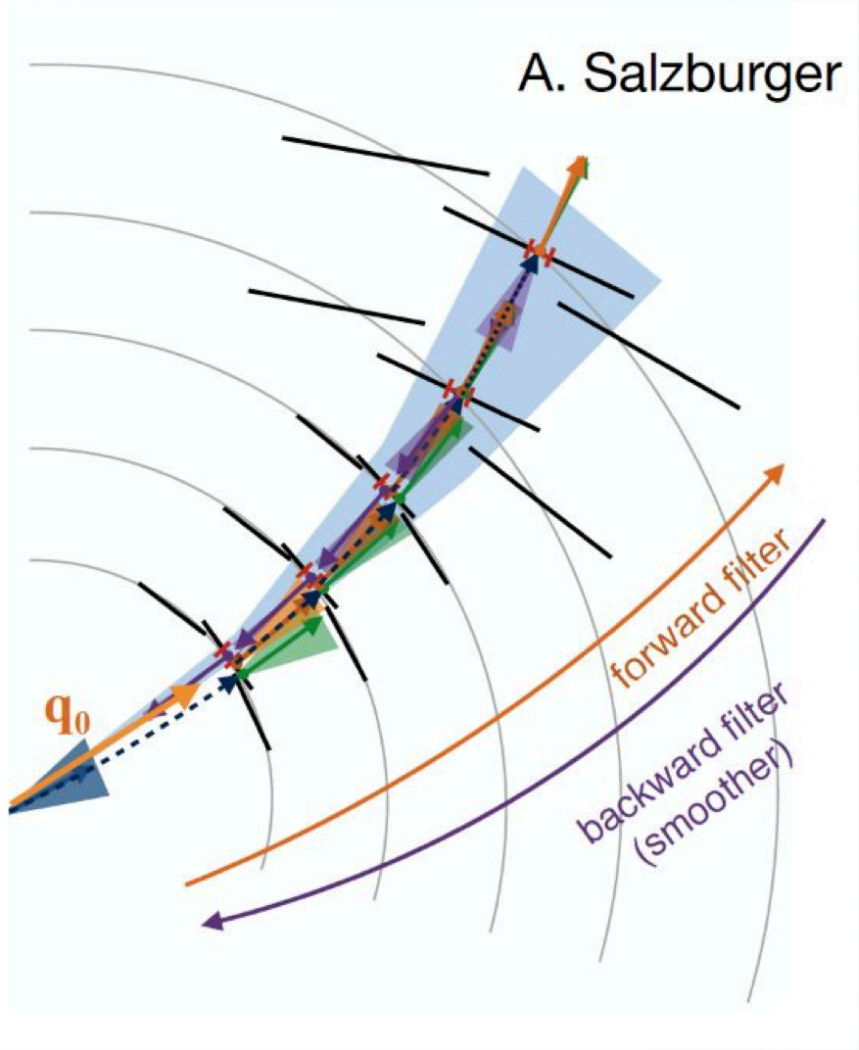
  ➡ Use **Kalman Filter (KF)** to account for these effects with and preserve a locally smooth trajectory





Kalman Filter in HEP

- Pioneered by P. Billoir and R. Frühwirth
- Progressive least-square estimation
- Equivalent to χ^2 fit if run with a smoother
- Start with track parameters and covariances to measurement surface and create predicted parameters, i.e. « **predicted** » state
- Combine predicted parameters with measurement to updated parameters, i.e. « **filtered** » state

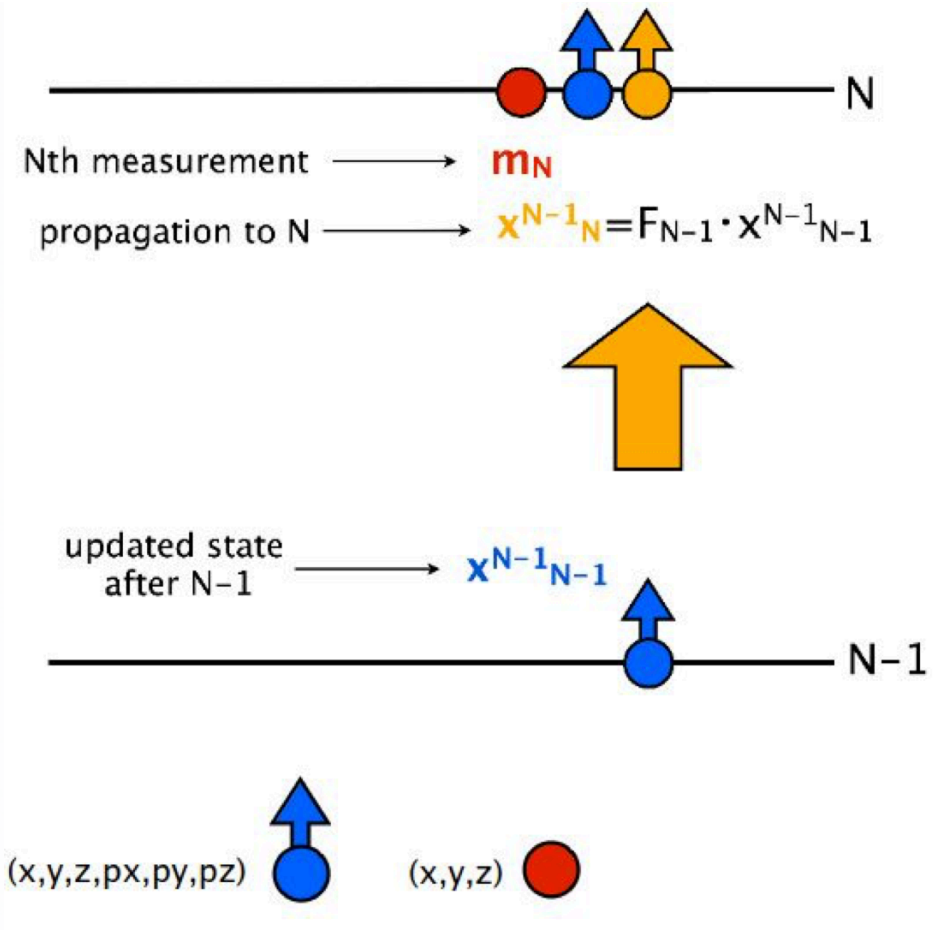


NIM A 262 (1987) 444 NIM A 294 (1990) 219

Kalman Filter for tracking

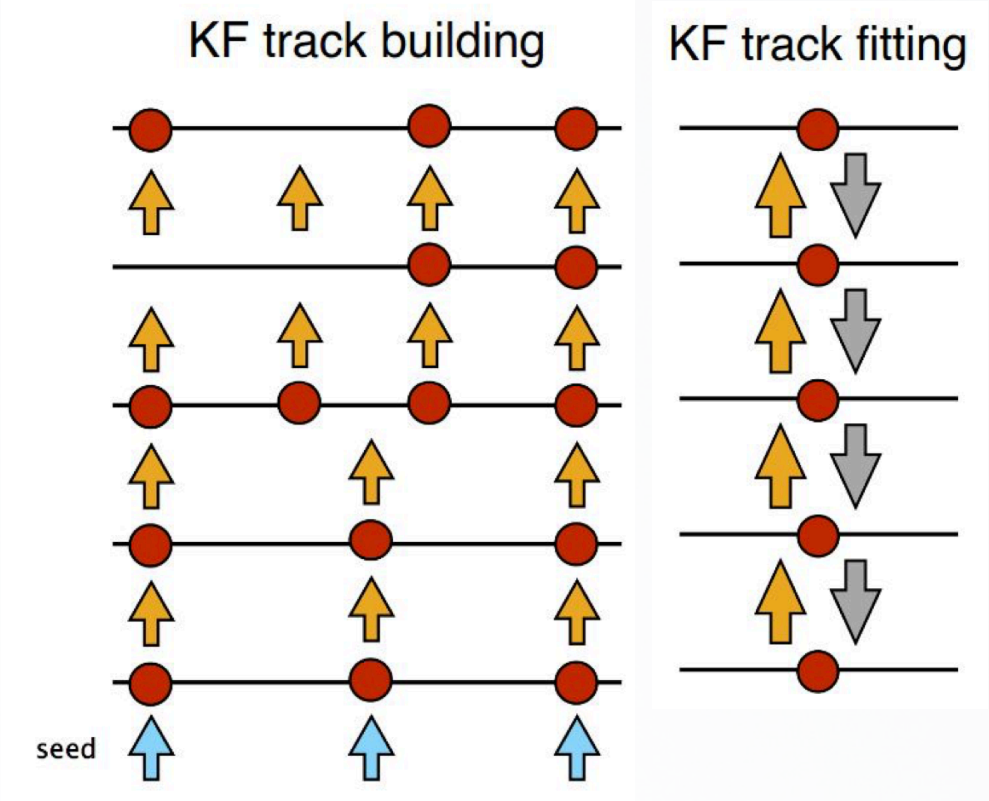
updated state after N ⟶ $x^{N}_{N} = x^{N-1}_{N} + K_N \cdot (m_N - H_N \cdot x^{N-1}_{N})$ ⟵ *essentially a weighted average*

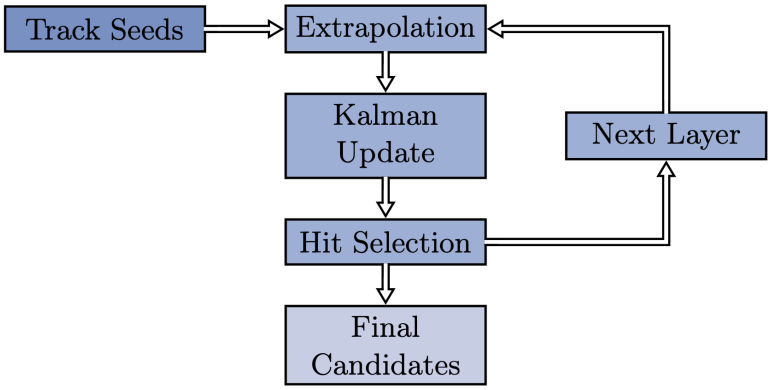
- The KF can be seen as an **iterative repetition** of the same logic unit
- After updating with the hit measurement, the state at layer N has smaller uncertainty than at layer N-1
- Reality: smeared by energy loss, multiple scattering, etc.



Defining trajectory

- Collect **additional detector layers** using initial estimate of the track parameters from seeding
- Form a complete track using **combinatorial KF (CKF)**; integrates pattern recognition and track fitting
- Most time consuming step - requires **branching** with potentially more than one track candidate per seed
- **Final fit** with a KF and a smoother





CkfTrajectoryBuilder

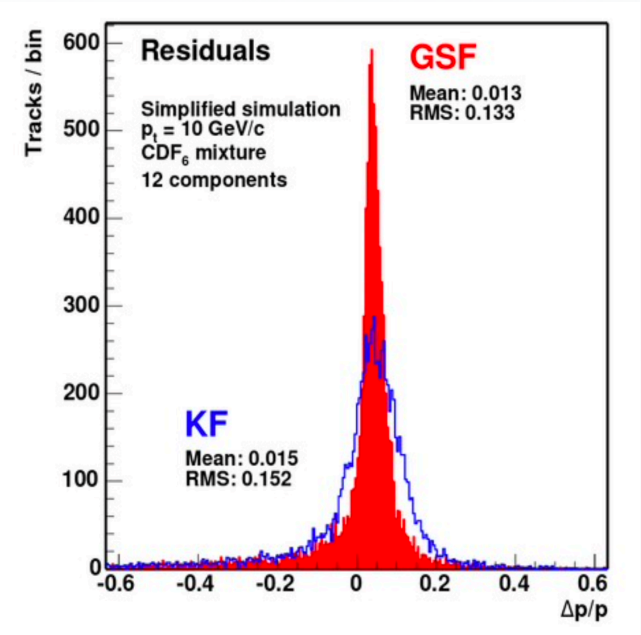
- Performs the pattern recognition (**trajectory building**) using a KF approach
  - ➡ Starts from the **seed layer** and uses the seed parameters as the initial state
  - ➡ Uses the navigation to **find compatible detector** layers that the trajectory is expected to intersect next
  - ➡ Uses the **propagator** to go to the surface of those layers
  - ➡ Searches for **compatible measurements** (hits) on these detectors, taking into account the angle of the trajectory with respect to the detector surface
  - ➡ **Updates** the trajectory state taking into account the added hit
- This is repeated until some **stopping criteria** are reached: no more detector layers, too many layers without compatible layers, etc.
- If multiple compatible hits are found on one layer, the **trajectory branches** are built
- Have to make sure to **limit** these branches, only retaining a certain number of trajectories at each step

Smoothening / Fitting

- The **TrackCandidates** produced in the trajectory building do not necessarily have the optimal track parameters yet (they can for instance be biased by constraints on the seeds from the seed building step)
- To find the **best fit parameters**, a final fit uses a KF and a smoother:
  - ➡ Start from the **seed layer** and use the seed parameters as the initial state
  - ➡ A **new initial track state** is obtained from the innermost hits of the track
  - ➡ It is then **propagated outwards**, updating the state with each hit sequentially
  - ➡ The final track state at the outermost hit is then used as the initial state for another round of KF going **outside-in**
  - ➡ The track state at each surface can then be obtained as the **weighted average of the two trajectories**, making maximal use of the available information
- During this procedure, the **hit position uncertainty** (and the hit position itself in the pixel detector) are updated using the track parameters
- An **outlier hit rejection** is performed based on χ^2 criterium, triggering a new filtering and smoothing if a hit is removed

GSF tracking

- KF for track reconstruction has its **limitations**:
  - ➡ It's a linear least-squares estimator and is optimal if all relevant PDFs are **Gaussian**
  - ➡ It uses **single-Gaussian distributions** to model the probability of energy losses in the detector material when propagating from layer to layer
  - ➡ **Not a good choice for electrons**, which have a high probability for large non-Gaussian energy losses due to bremsstrahlung
- For electrons we use a non-linear extension of the KF, the **Gaussian Sum Filter (GSF)**:
  - ➡ Approximates the energy loss distribution with a **mixture of several Gaussians**, based on the Bethe-Heitler model
  - ➡ This improved modeling of radiative energy losses leads to an **improved resolution** for track parameters



Track selection

Track selection

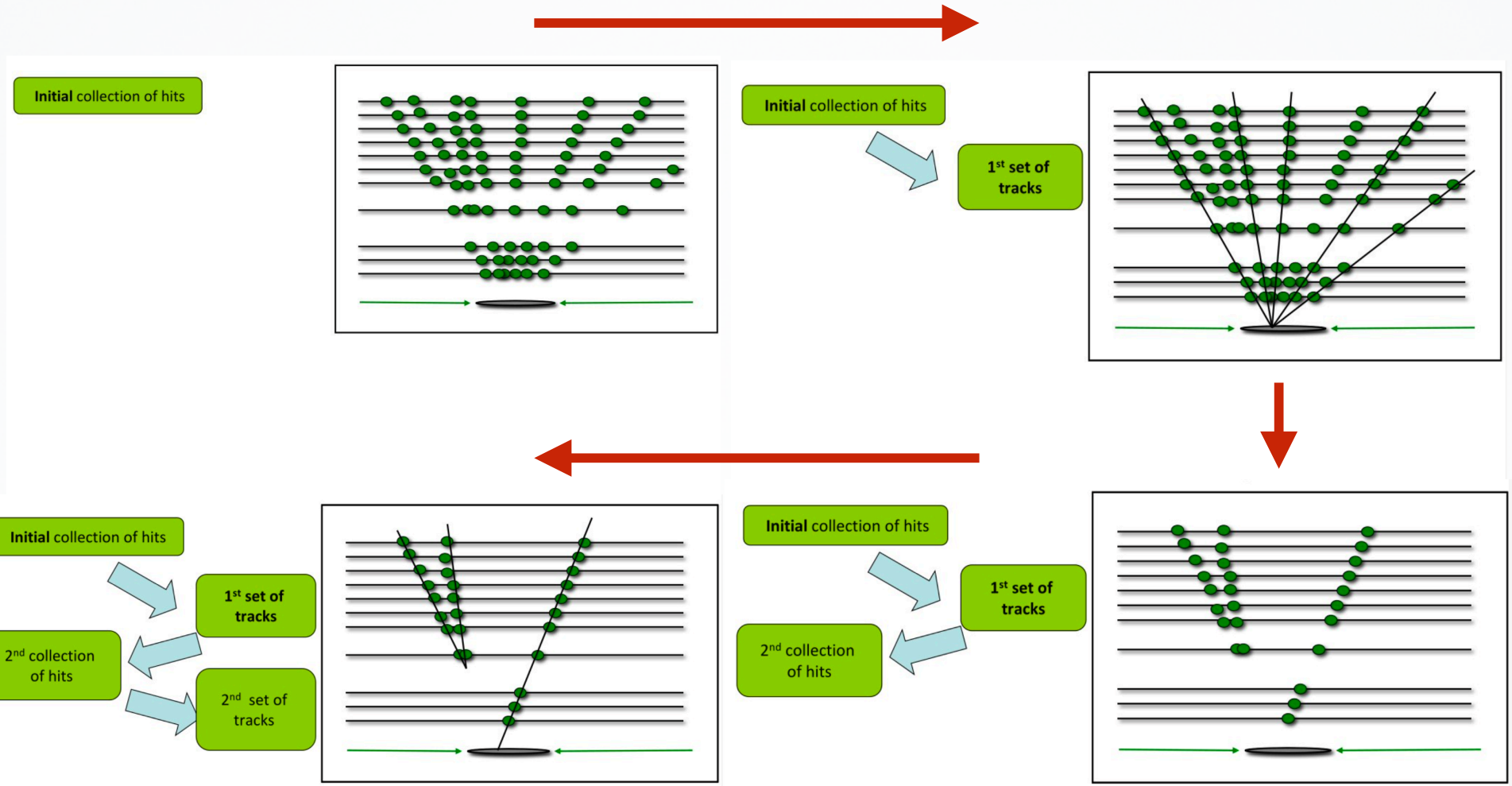
- Tracks are **selected** based on their hit pattern (number of hit in the pixels and strips, number of lost hits, number of 3D hits, etc.), impact parameters, χ^2 of the fit, etc.
- **Optimal cuts** obtained for each track iteration using a BDT
- Selection is then applied cut-based using the **MultiTrackSelector**
- Three working points: **Loose**, **Tight**, **HighPurity**
- Only **high-purity** tracks are actually used in most cases

Iterative tracking

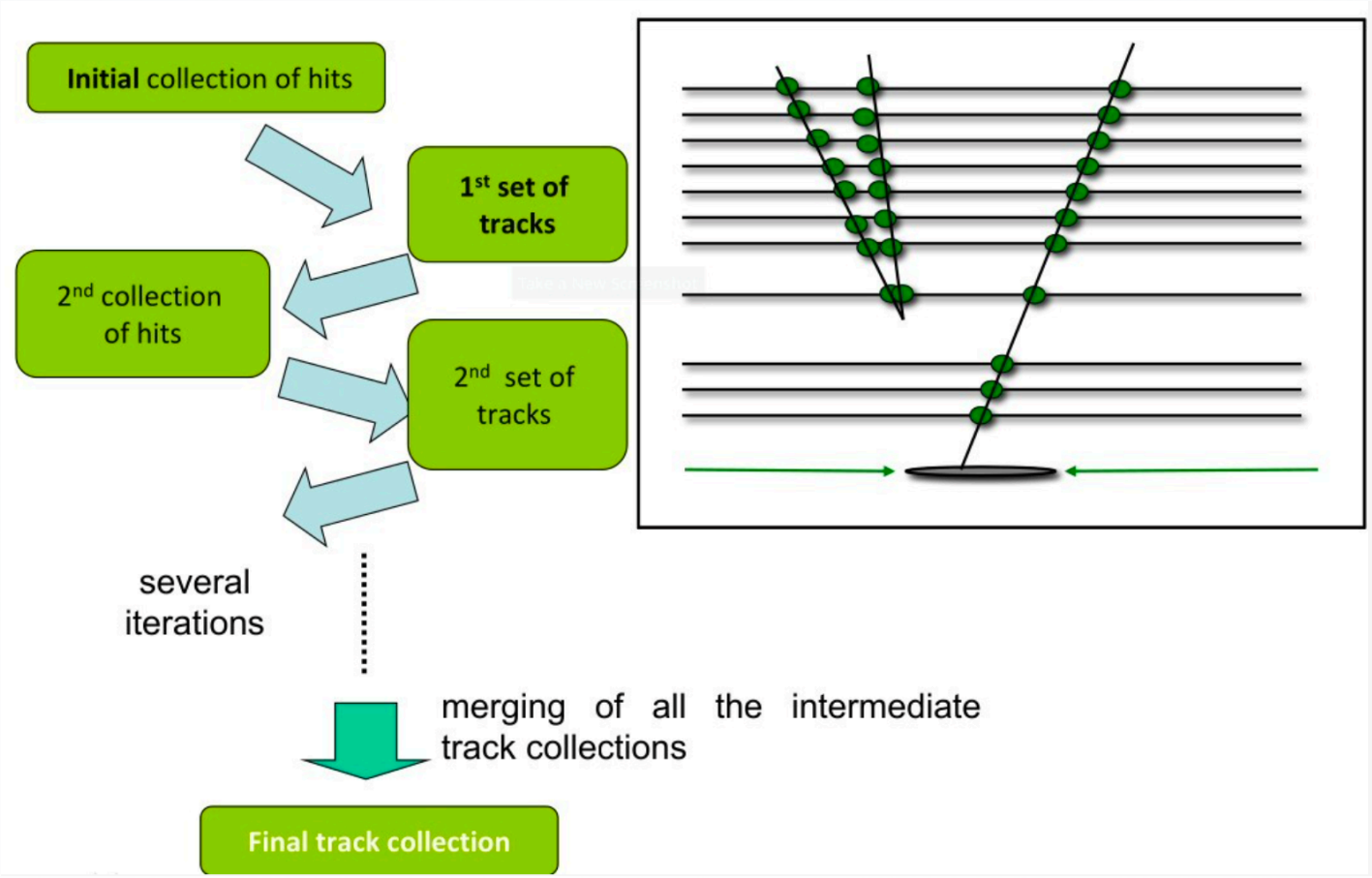
Iterative tracking

- In general, **high-momentum primary tracks** are **easy to reconstruct** and it does not take too much time
  - ➡ Primary vertex and beamspot constraint
  - ➡ High-precision pixel hits give high-quality seeds
  - ➡ Less multiple scattering, smaller search window
- **Relaxing** these requirements **increases the combinatorics**
- Still **very useful** (yet difficult) to reconstruct low-p_T, primary tracks missing a pixel hit, displaced tracks
  - ➡ Conversions, nuclear interactions, heavy-flavor decays
  - ➡ Tracks crossing inefficient parts of the pixel detector
- **Iterative tracking** aims at **reducing the combinatorial problem** so that problematic tracks can also be reconstructed with the CPU time budget
- The idea is to **run track reconstruction several times**, and each time hits that are used by tracks from previous iterations are masked

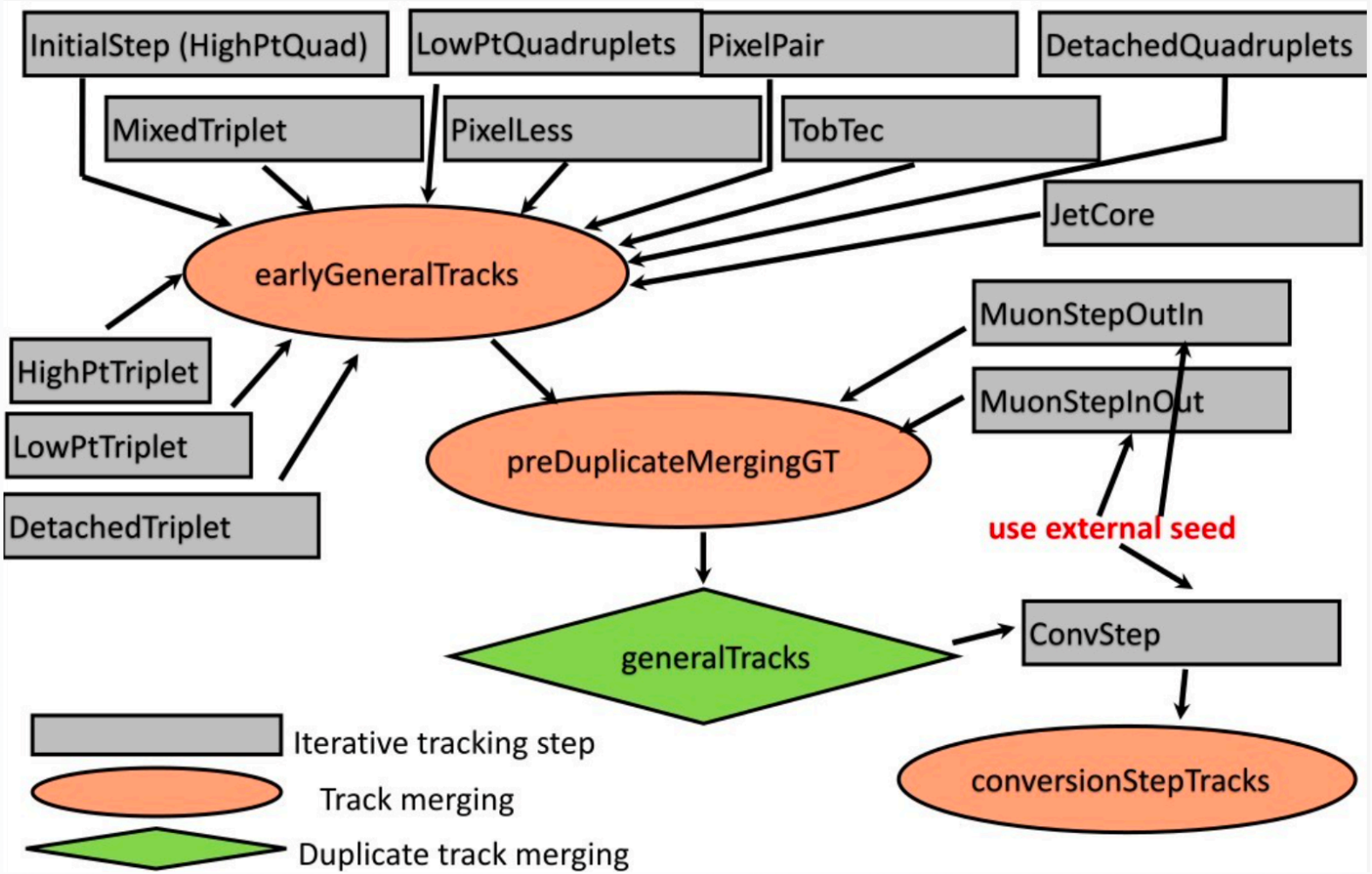
Iterative tracking



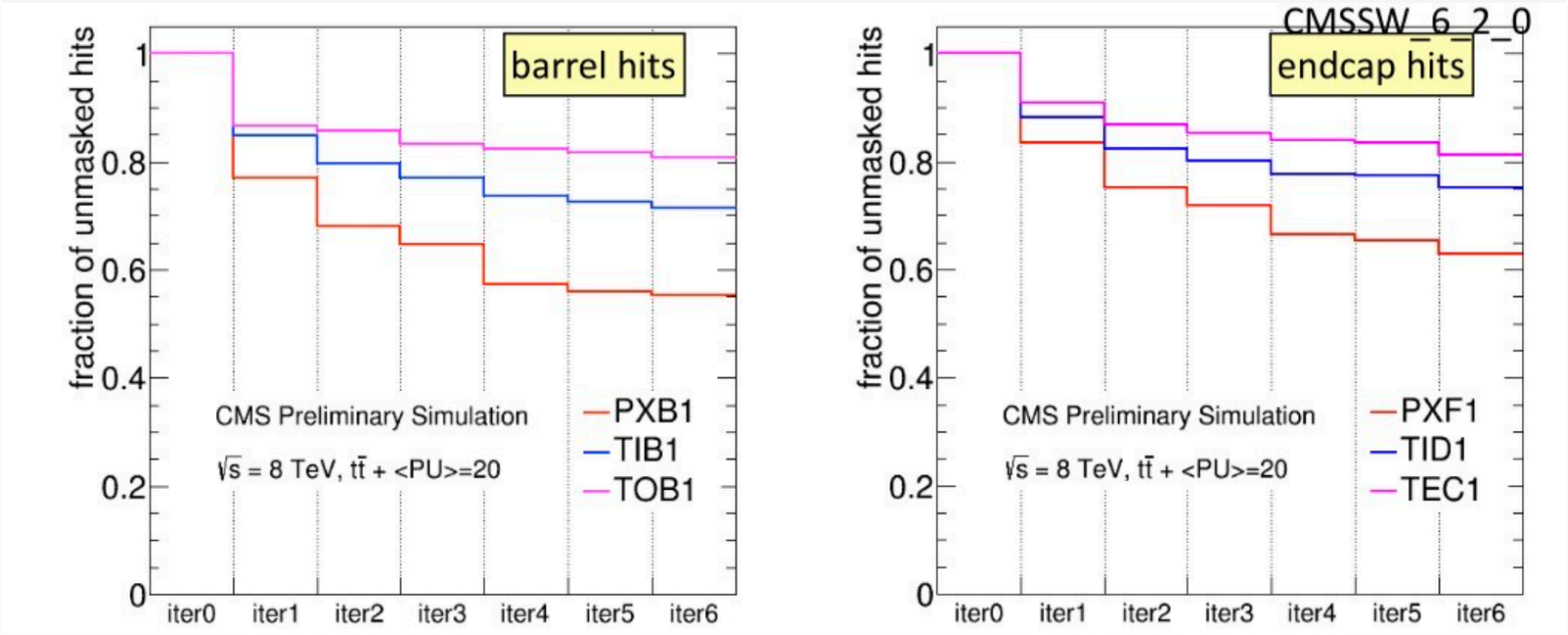
Iterative tracking



Iterative tracking sequence



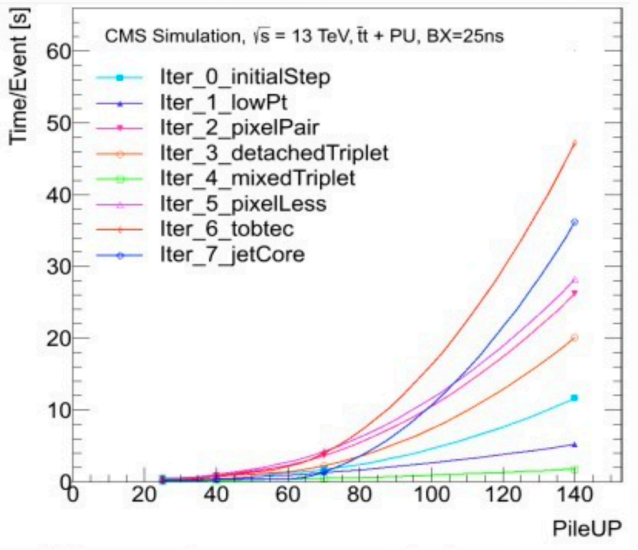
Performance of the iterative tracking

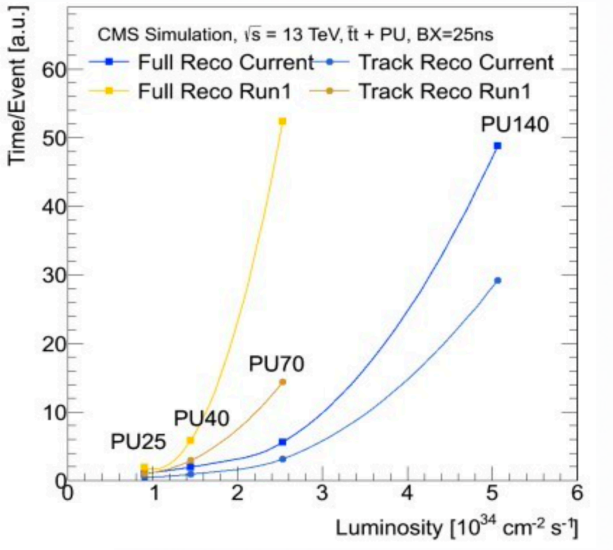


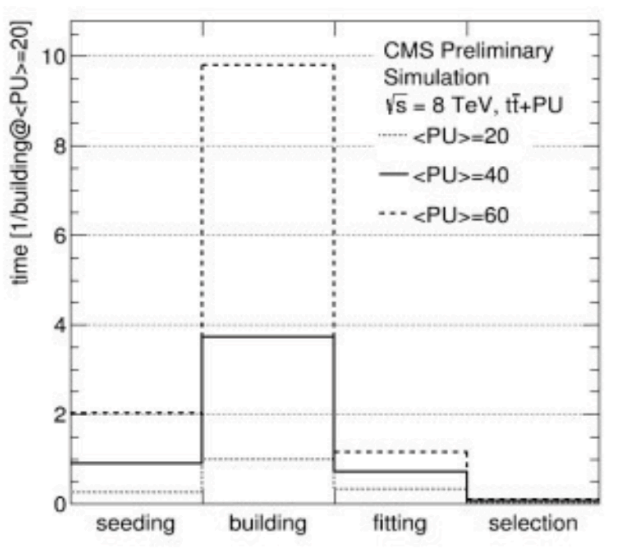
- **First iteration** is most efficient in reducing the number of hits
- **After all iterations**, more than half of the hits are still not associated to tracks
- The iterative tracking approach **reduces the combinatorics** but tracking is **still a big challenge**

Timing

- Tracking takes **most of the computing time** of CMS event reconstruction
- Timing **grows with pileup**, especially above 70
- Most of the time is spent during pattern recognition (**building**)
- Steps **not using pixel quad&triplets** are those taking most of the time (tobTec, pixelLess, pixelPair)







Geometry

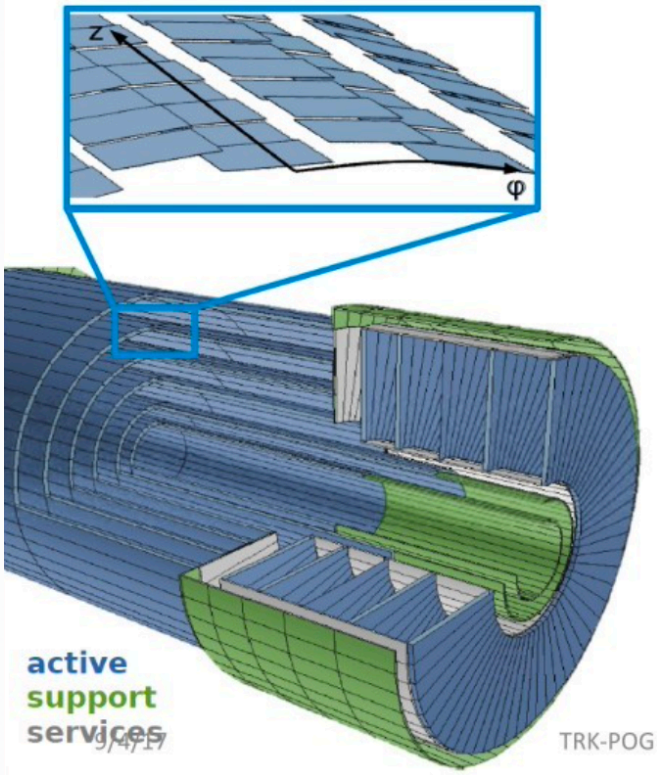
Layout modeling

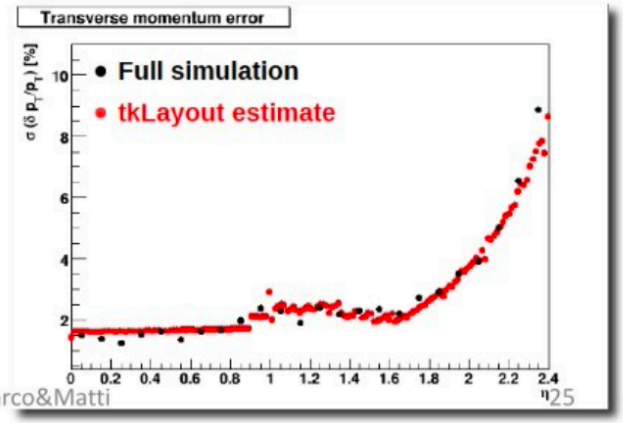
layout modelling

Design of detector: a many-parameters problem

Innovative solution: **tkLayout**

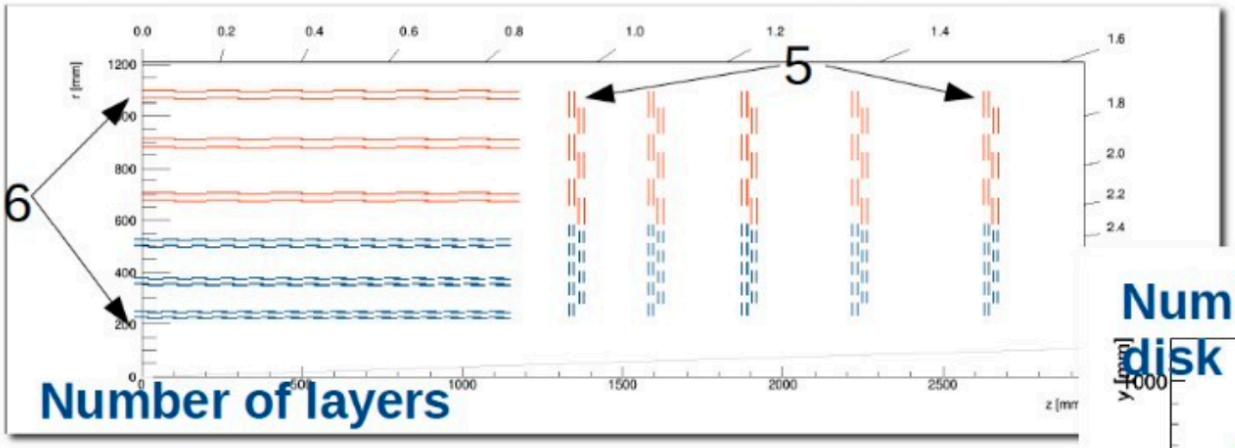
- detector layout generator software using a **simple** description of design parameters
- creates a 3D model of the layout with a description of the materials
- makes an *a priori* estimate of **tracking resolution** (much faster than a traditional simulation)
- **output was validated** by comparing with full simulation of present detector

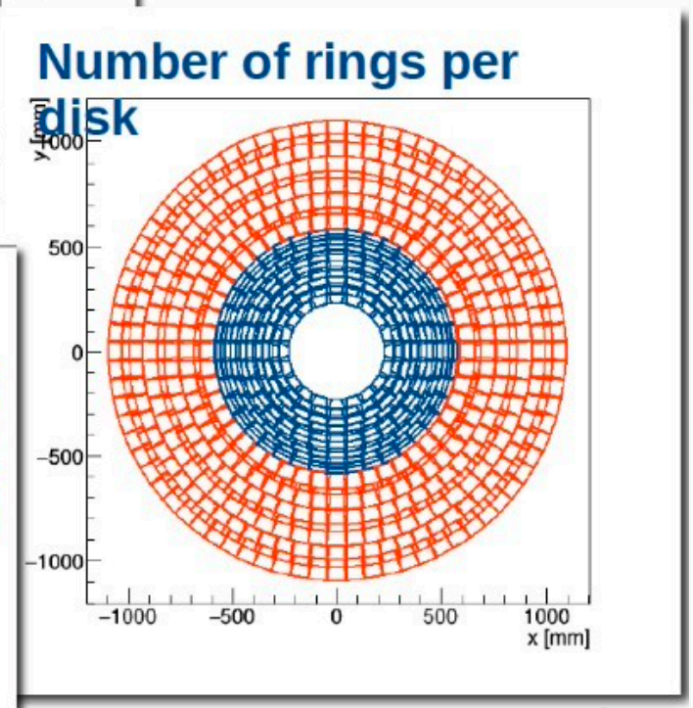


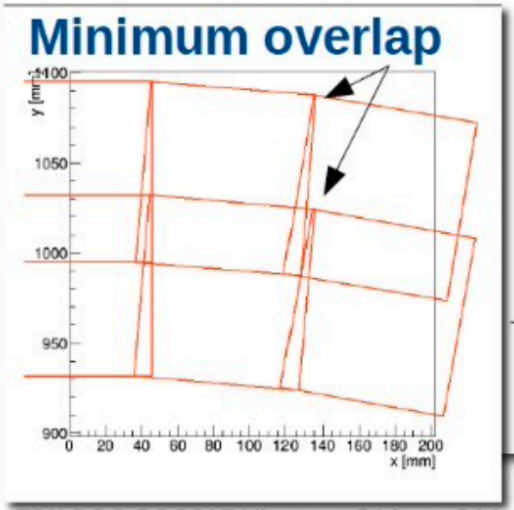


Detector geometry

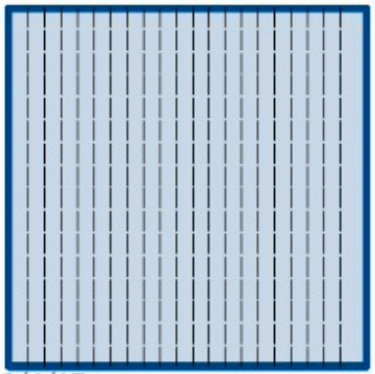
Simple parameters define the entire geometry, like







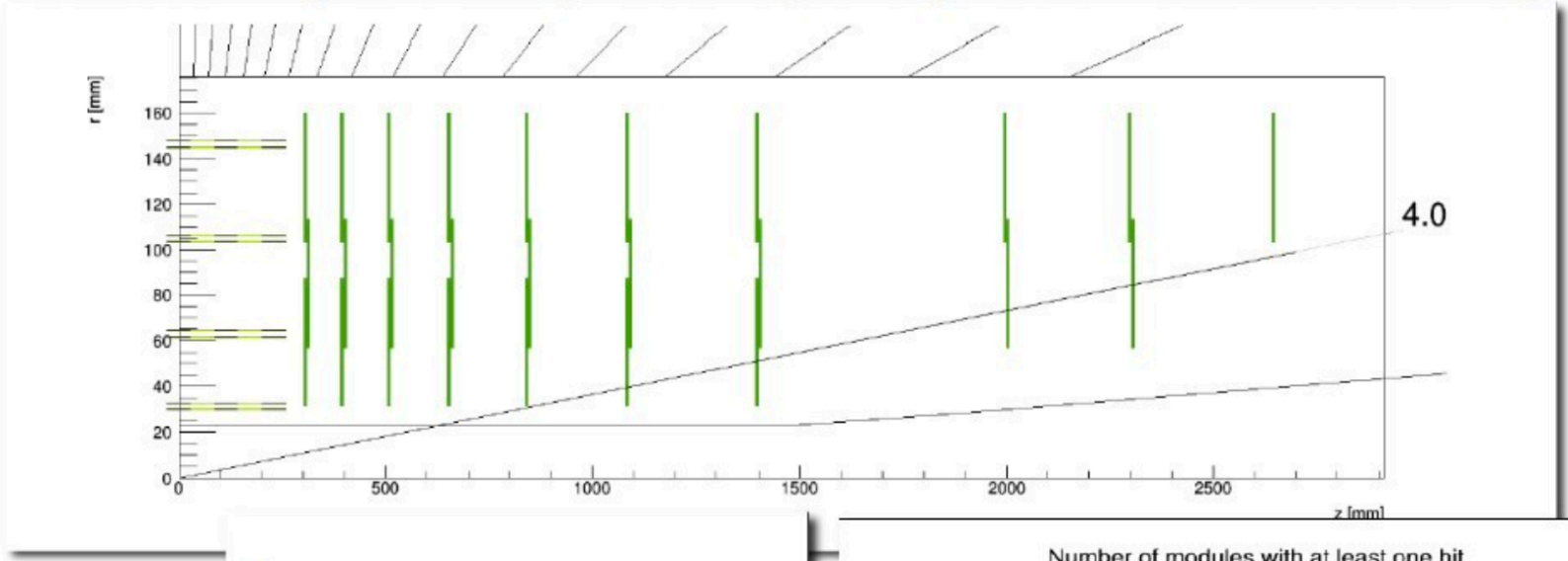
Sensor geometry



(ex. 1024×16 MacroPixel)

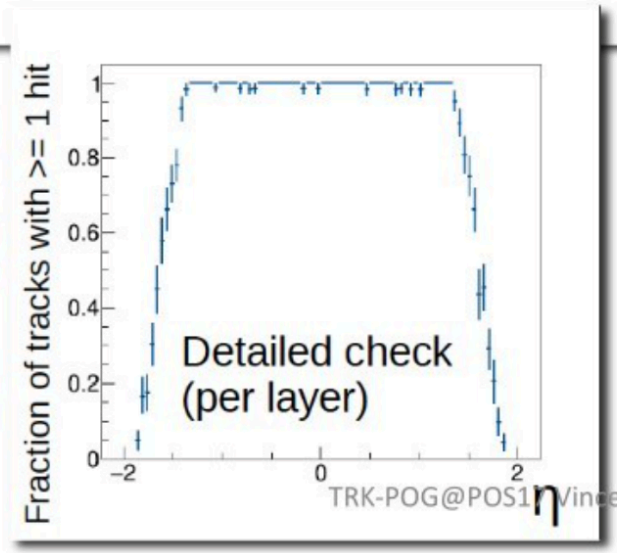
Coverage check

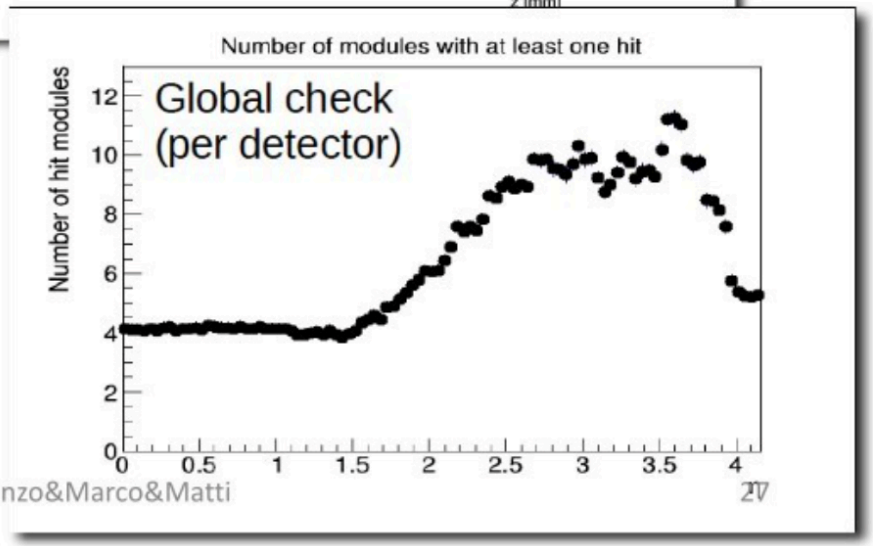
To validate geometry and qualify hermetic coverage



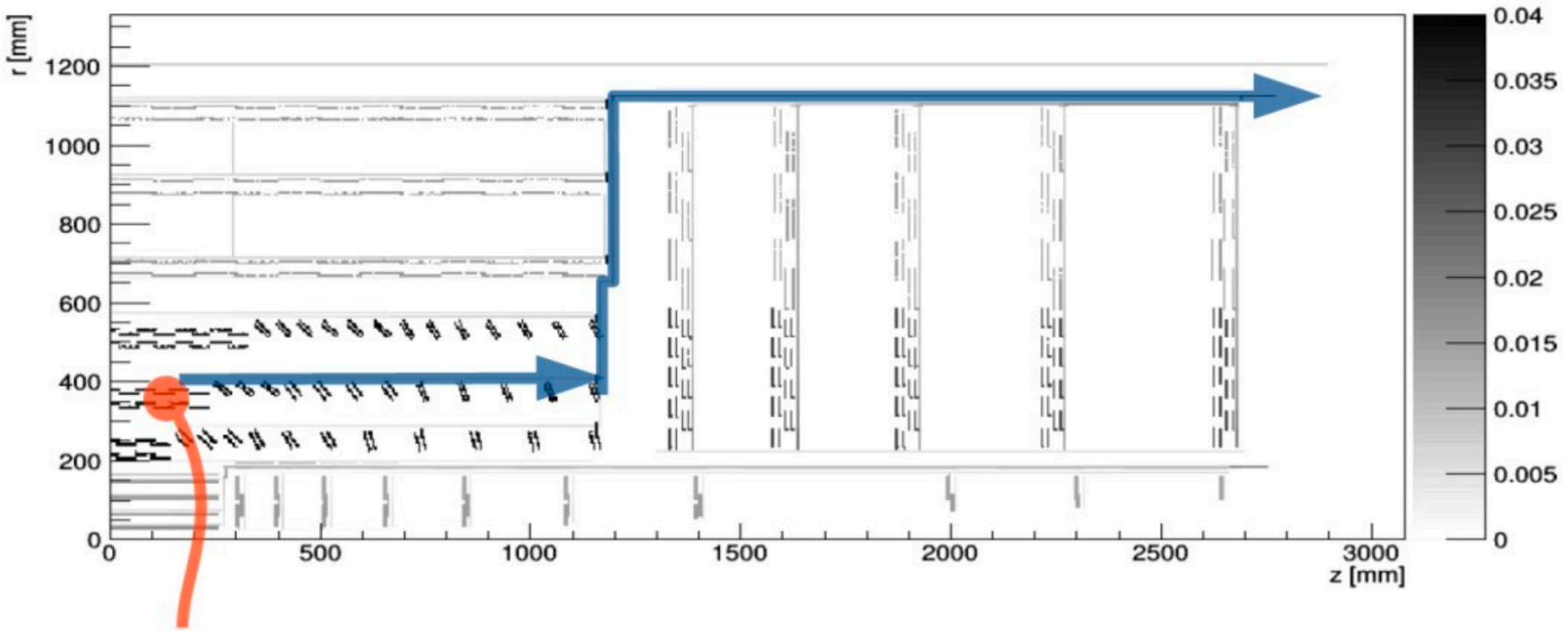
Beam spot shape and size are taken into account

"Tracks" all have $p_T=\infty$





Material

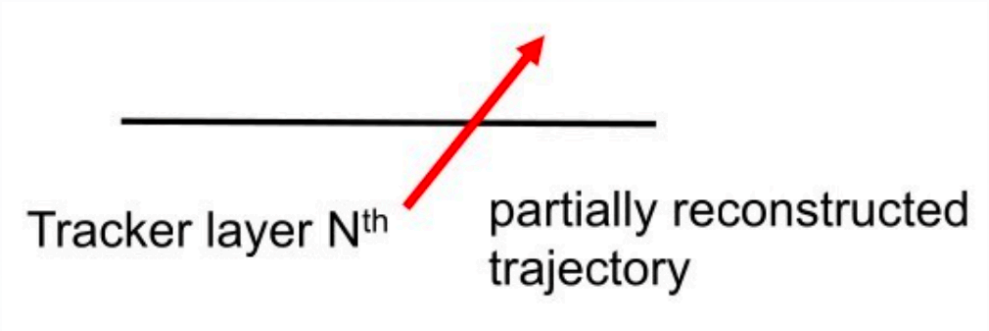


Material on active elements + Material for services automatically routed

Navigation

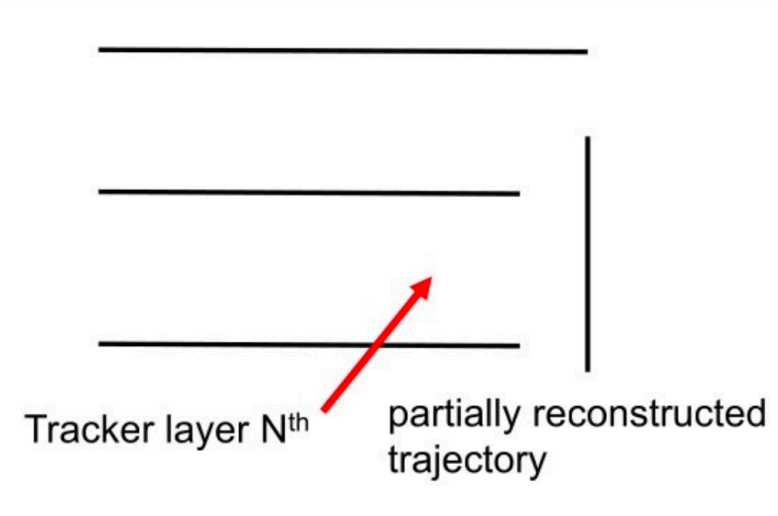
Trajectory builder

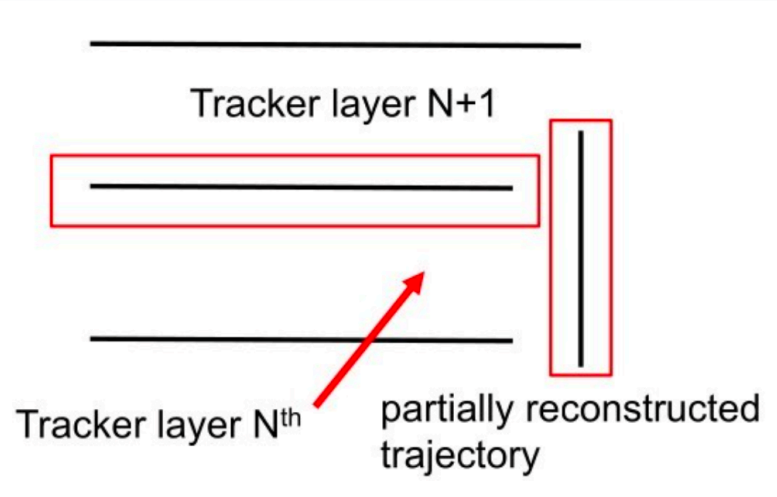
Input: trajectory state on the Nth layer,
TrajectoryStateOnSurface



Trajectory builder

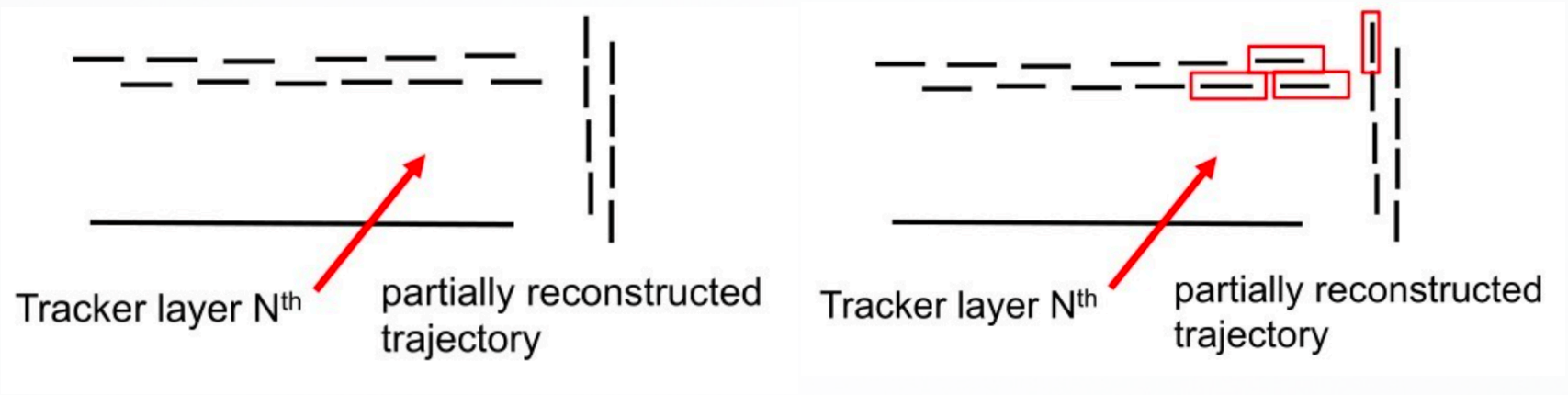
Input: trajectory state on the Nth layer
→ find **next** compatible layer (or layers)





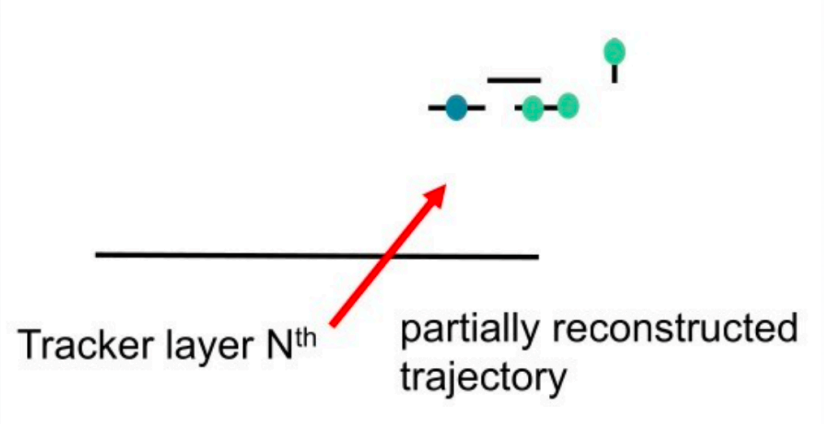
Trajectory builder

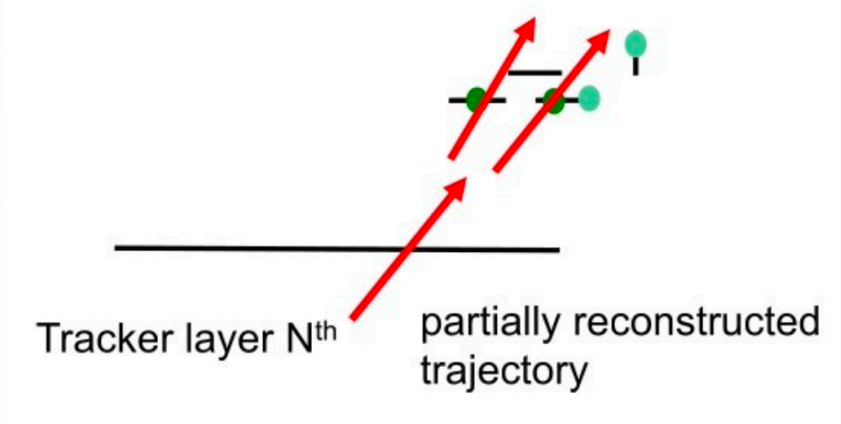
Input: trajectory state on the Nth layer
→ find **next** compatible layer (or layers)
→ find list of compatible **detectors** on layers N+1



Trajectory builder

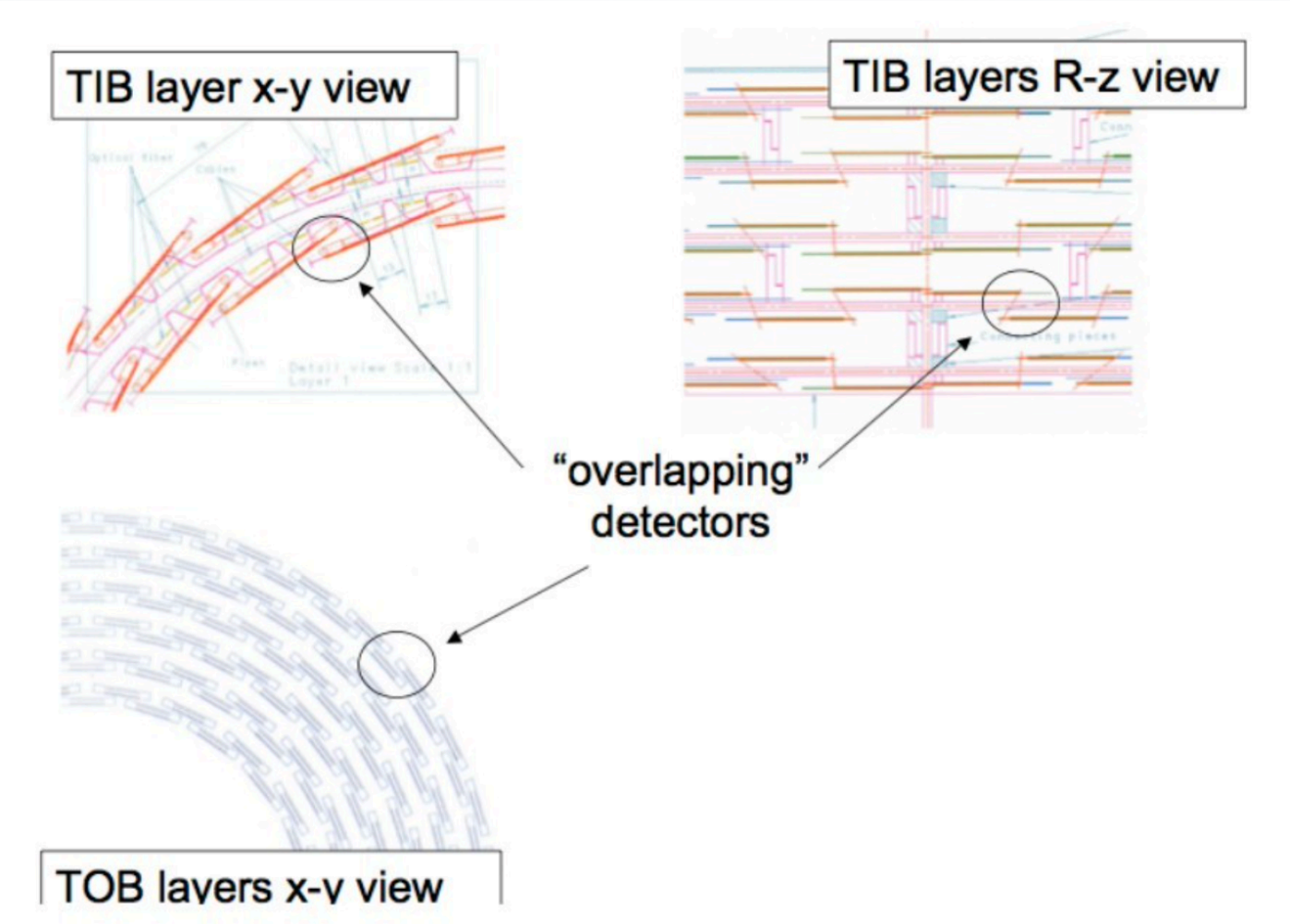
Input: trajectory state on the Nth layer
→ find **next** compatible layer (or layers)
→ find list of compatible **detectors** on layers N+1
→ find list of compatible **hits** on compatible detectors
→ create **new trajectory** with updated state for each compatible hit



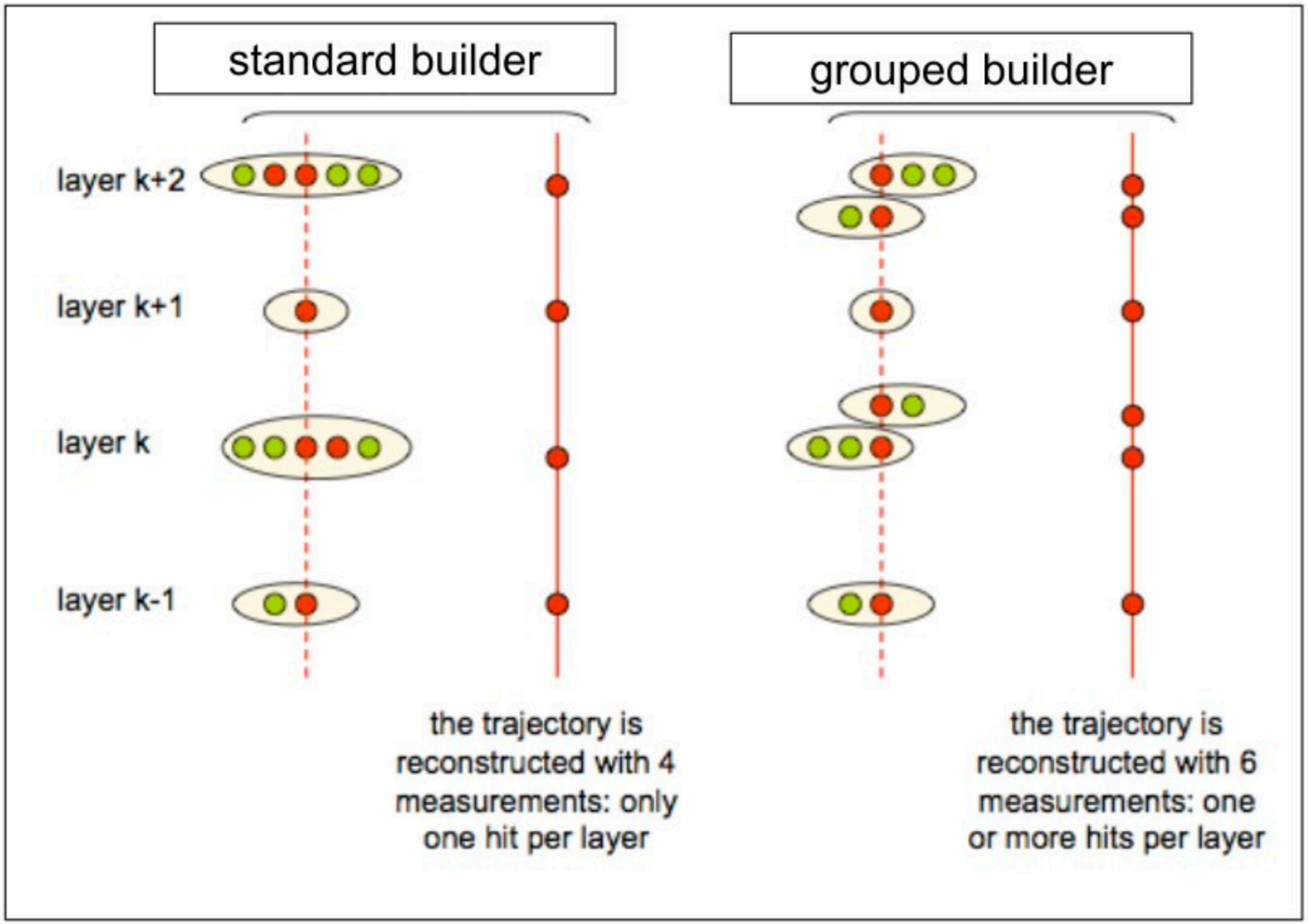


Output: two new states will be projected onto the next Tracker layer

Standard vs grouped builders

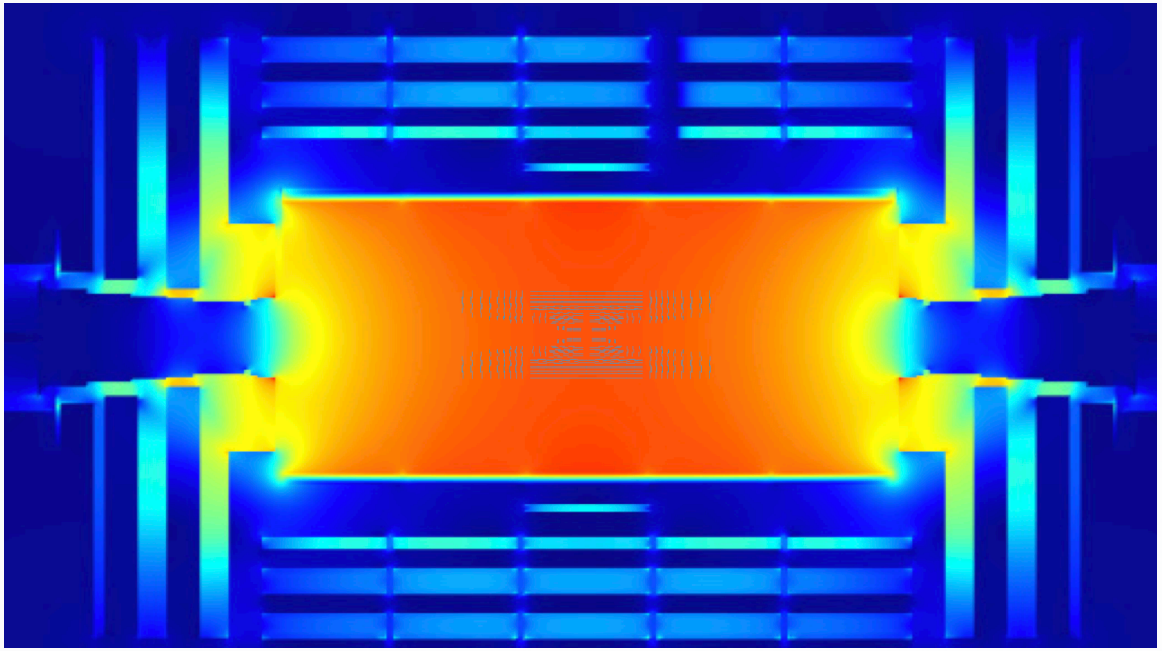


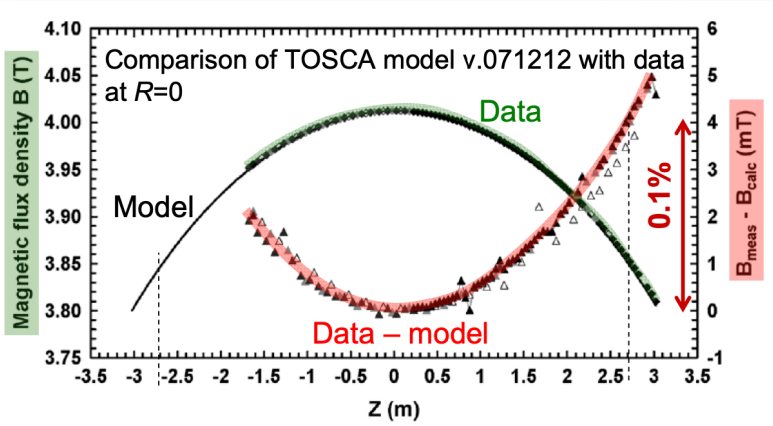
Standard vs grouped builders

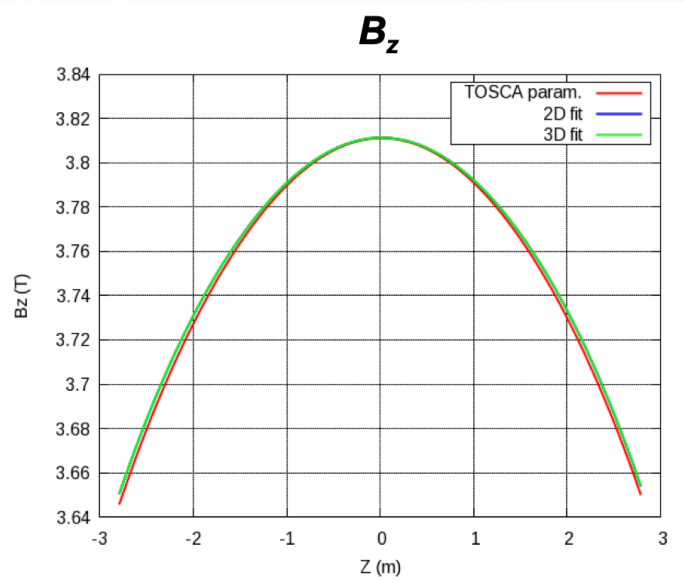


Propagation

Magnetic field







Full and « **parabolic** » (fast) approximations
using 2D/3D fits

Propagators

- **Propagators** are needed to **extrapolate a trajectory** from one surface (or space point) to another, taking into account the magnetic field and material effects including energy losses
- Tracking generally makes use of **three propagators**:
  - ➡ AnalyticalPropagator
  - ➡ PropagatorWithMaterial
  - ➡ RungeKuttaPropagator

Propagators

- **AnalyticalPropagator**:
  - ➡ Calculates analytically the intersection between a helix and detector plane or cylinder
- **PropagatorWithMaterial**:
  - ➡ Builds on the AnalyticalPropagator and adds energy losses in material (Bethe-Bloch/Bethe-Heitler) and accounts for multiple scattering
  - ➡ Used for most steps of the track reconstruction, except the final fit
- **RungeKutta**:
  - ➡ Uses Runge-Kutta methods for iteratively solving first order differential equations (such as particle in a B-field)
  - ➡ Can take into account the non-uniformities in the magnetic field
  - ➡ Used in the final track fit
  - ➡ More precise but slower

Additional propagators

- **SteppingHelix**:
  - ➡ Designed for volumes containing lots of material
  - ➡ Used in the muon reconstruction to propagate through the calorimeters and the magnet yoke
- **GEANE**:
  - ➡ Propagator and fitted developed for GEANT3 in the early 90s
  - ➡ Used in several previous experiments
  - ➡ Very precise but very CPU-intensive

Links

- NanoAOD content documentation: https://cms-nanoaod-integration.web.cern.ch/autoDoc/
- Tracking performance with in Run 1: https://arxiv.org/abs/1405.6569
- Track reconstruction with mkFit in Run3: https://cds.cern.ch/record/2814000
- SW guide documentation: https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideTrackReco, https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideVertexReco
- Tracking Training Day 2019: https://indico.cern.ch/event/849864
- Tracker Training Day 2023: https://indico.cern.ch/event/1238081
- Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors (R. Frühwirth & A. Strandlie) https://link.springer.com/book/10.1007/978-3-030-65771-0
- Tracking POG webpage: https://cms-tracking.docs.cern.ch/
- Tracking POG CMS Talk: https://cms-talk.web.cern.ch/c/physics/trk/148