tracking at CMS

mia tosi
on behalf of the CMS collaboration

Dubrovnik, Apr 30th – May 3rd, 2019

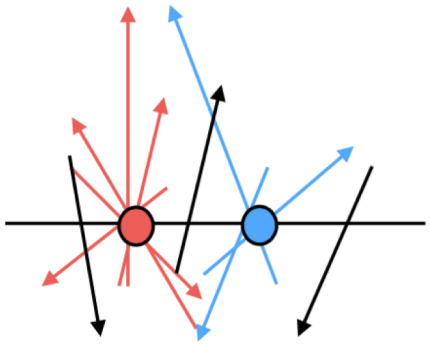
-- Game of Flavours – CMS Heavy Flavour Tagging Workshop 2019 --

introduction

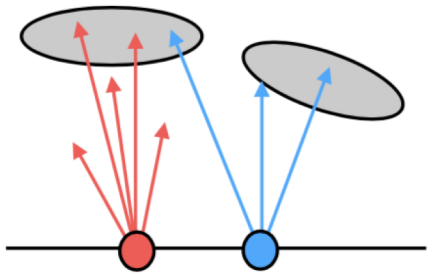
tracking is a key ingredient of reconstructing the full event

the tracking challenge at the LHC

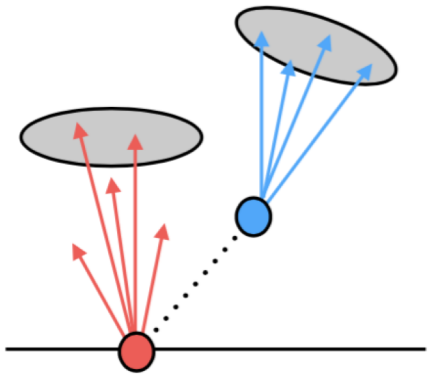
- typically 30 charged particles w/in the tracker acceptance per proton-proton collision and 40 collisions per event: O(1200) charged particles per event
- these need to be reconstructed w/
  - very high efficiency (>90%)
  - precise track parameters
  - very low fake rate: O(~%)
  - quickly (stringent CPU limits)
- very strong requirements on track reconstruction algorithms
- track reconstruction is not just about reconstructing charged particles: used in almost every element of reconstruction



vertex reconstruction



pile up removal

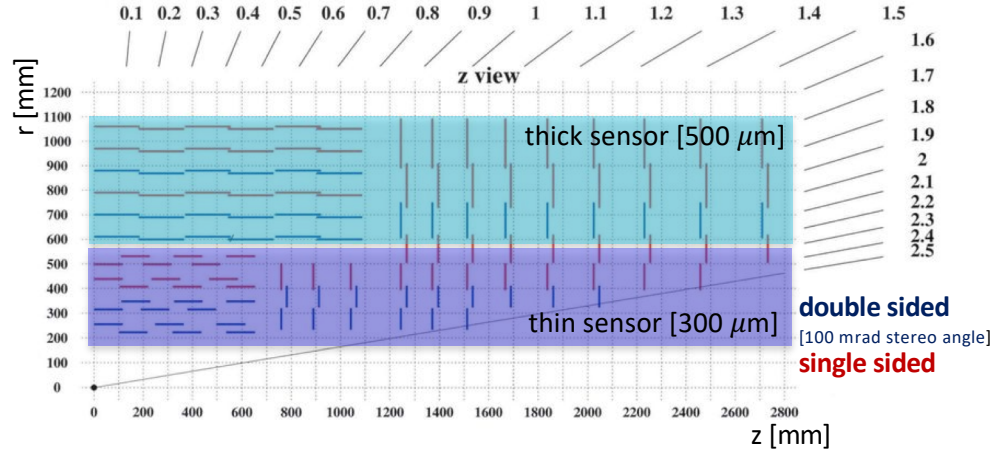


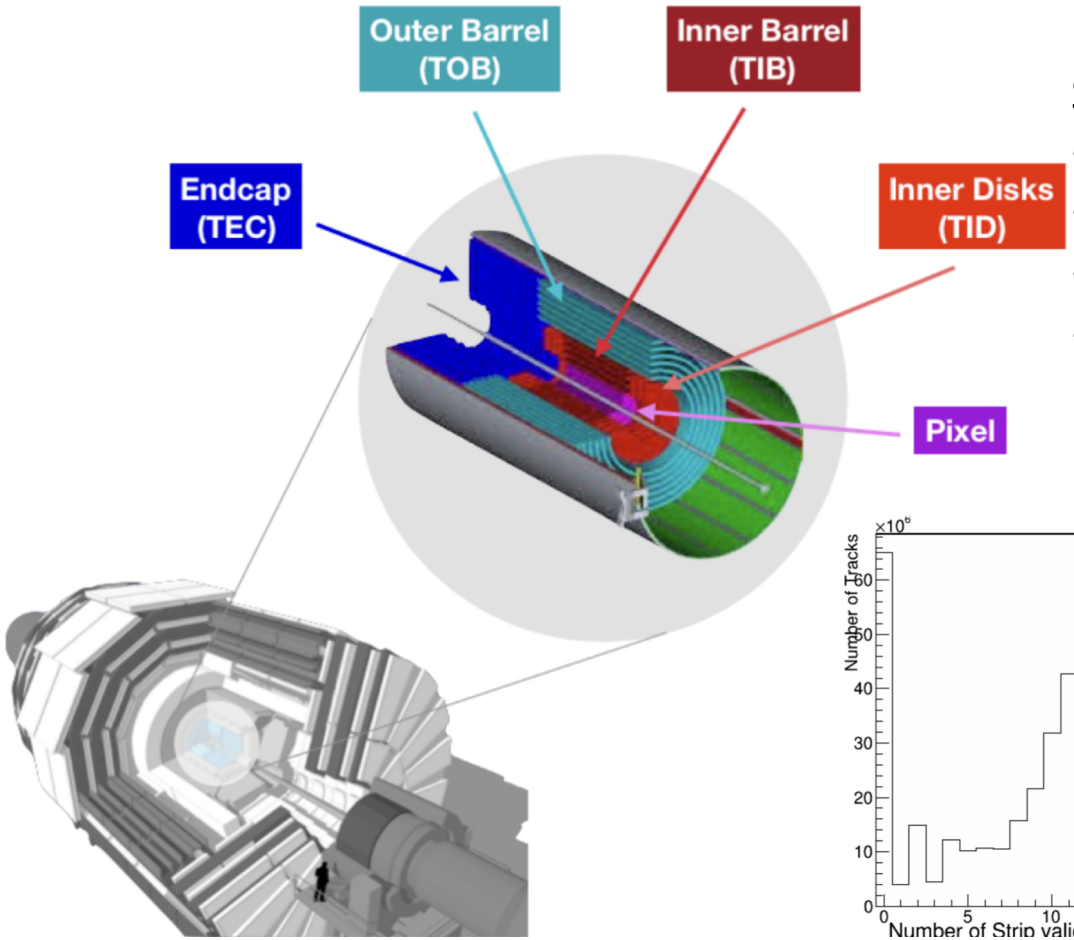
jet flavour tagging

see Petra and Chris' slides
for details on Flavour Tagging approaches in ATLAS and CMS

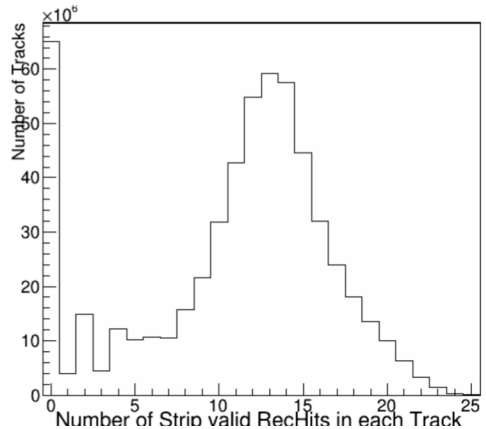
CMS tracker: Silicon Strips

- *O(10 M)* strips
- *O(200) m²* of sensors
- hit resolution: (10,40) x (230,530) *μm*
- occupancy: *1-3%*
- coverage up to $|\eta| < 2.5$





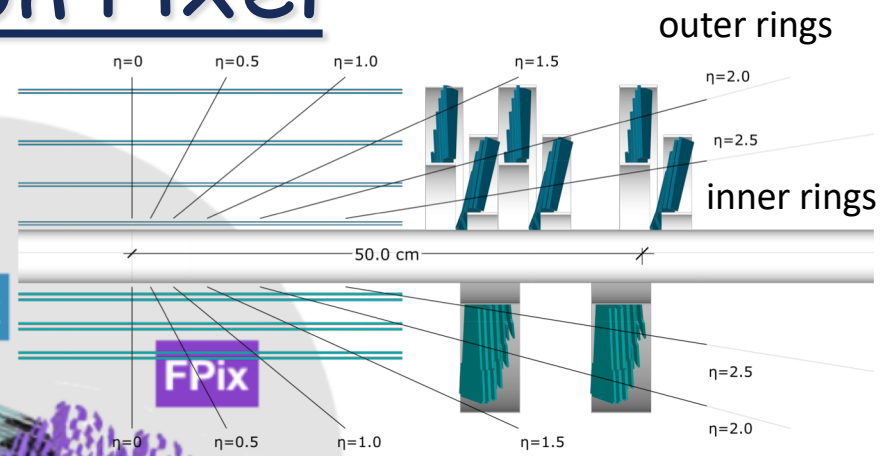
Sub-detectors
- Inner Barrel (**TIB**): 4
- Inner Disks (**TID**): 3 (x 2)
- Outer Barrel (**TOB**): 6
- Endcap (**TEC**): 9 (x 2)

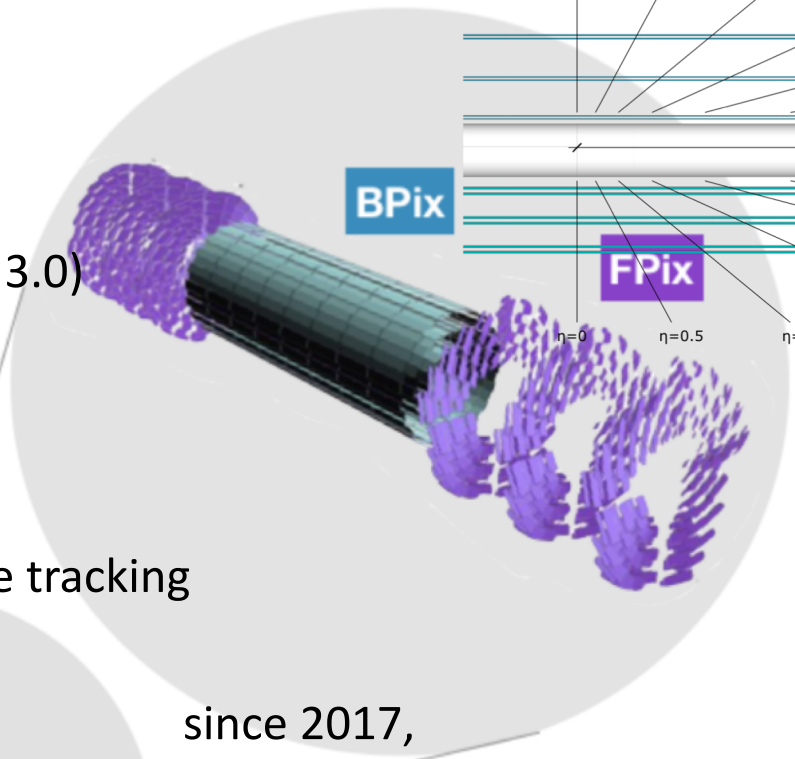


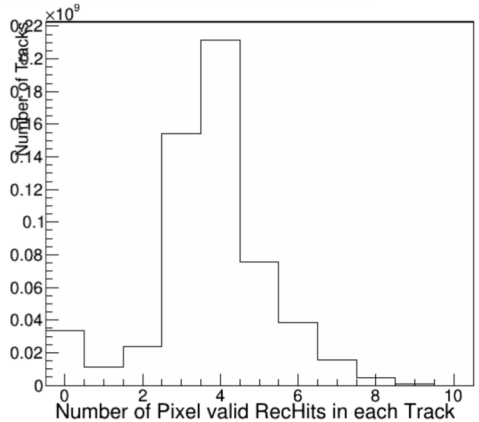
many layers:
redundancy
12 hits per track on average

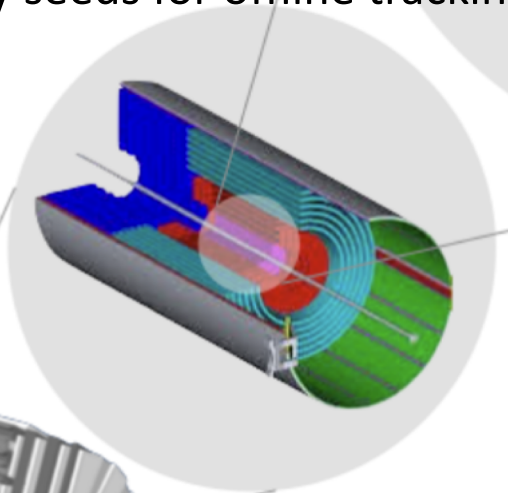
CMS tracker: Silicon Pixel

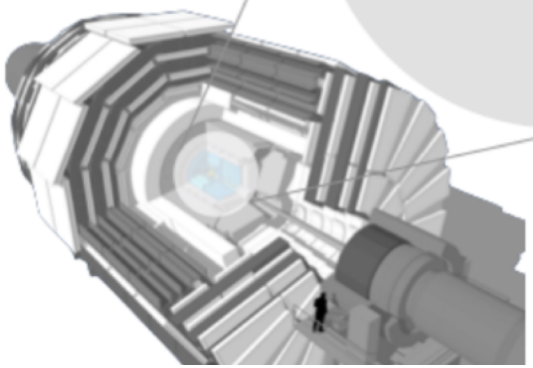
- *127 M* pixels,
- 100 x 150 μm^2 in size
- hit resolution: 10 x (20,40) μm
- occupancy: 10^{-3}
- coverage up to $|\eta| < 2.5$ (even 3.0)
- layer position [cm]:
  - BPix: 2.9, 6.8, 10.9, 16.0
  - FPix: 29.1, 39.6, 51.6
- high segmentation:
  - high quality seeds for offline tracking











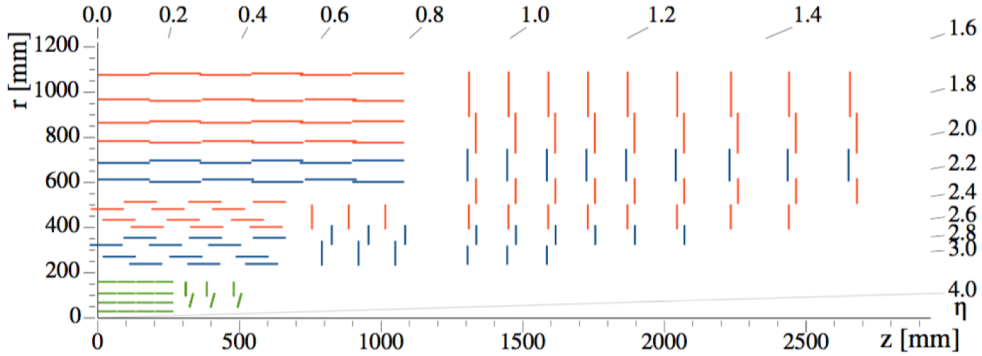
since 2017,
one additional tracking point,
- in both barrel and forward regions:
  - 4-hit seeds
  - lower fake rate!

smaller radius of the innermost pixel layer
- closer to the interaction region
- improve tracking and vertexing performance

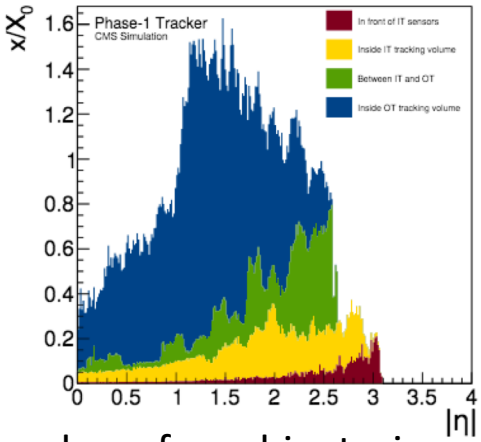
reduced material budget
- reduce multiple scattering

track reconstruction in CMS

- few, but precise measurements
- non negligible amount of dead material inside the tracker volume

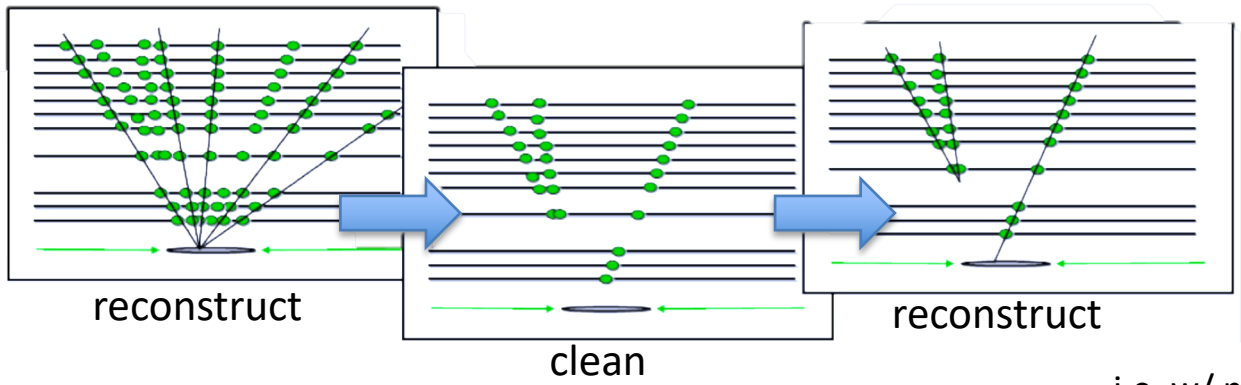


R_{inner} [cm]	R_{outer} [m]	$\|\eta\|$ coverage	B field [T]	X_0 @ $\|\eta\|=0$	p_T resolution @1 (100) GeV, $\|\eta\|=0$	d_0 resolution @1 (100) GeV, $\|\eta\|=0$ [μm]
3	1.1	3.0	3.8	0.4	0.7 (1.5)%	90 (20)



main tracking algorithm : Combinatorial Track Finder used in iterative steps

- limits the number of combinatorics in pattern recognition
- tracking reach guarantee, w/o degrading computing performance



high-quality tracks are reconstructed first,
their hits are removed,
other (more difficult) tracks are reconstructed
from the remaining hits

i.e. w/ more multiple scattering, loops, displaced tracks..

track reconstruction in CMS

in each iteration, tracks are reconstructed in four steps:

1. seeding: use combination of pixel, strip or mixed hits

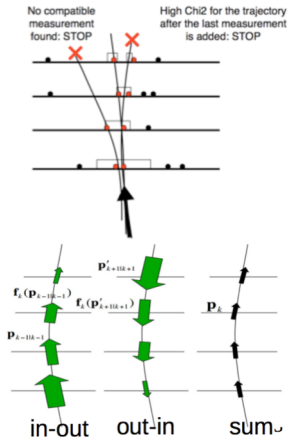
provides track candidates,

with an initial estimate of the trajectory parameters and their uncertainties

2. pattern recognition: alignment uncertainty taken into account

hit compatible w/ predicted track position are added (Kalman update)

to the trajectory track parameters are updated



3. final fit: taking into account the Field non uniformity and a detailed description of the material budget

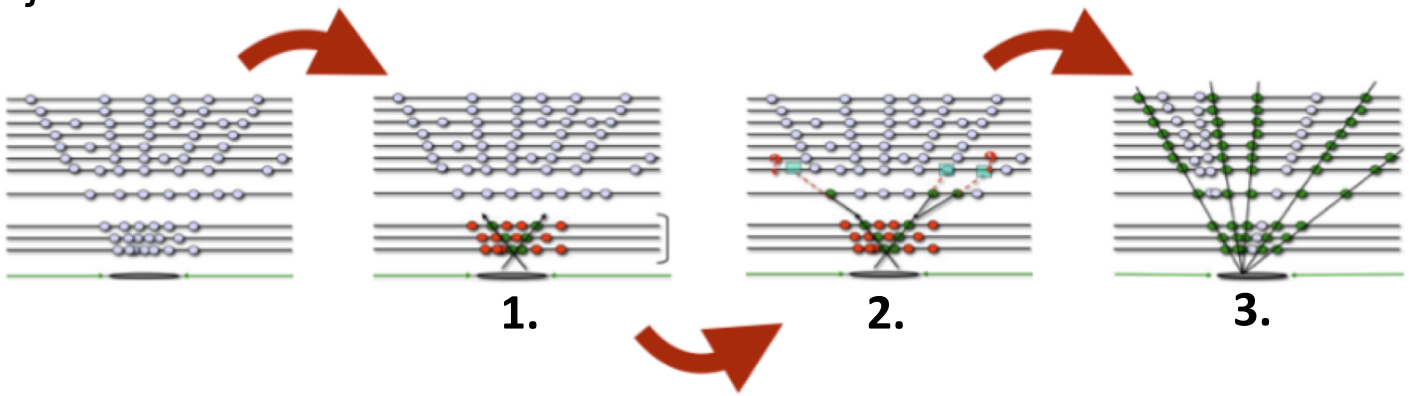
provides the best estimate of the parameters of each smooth trajectory

after combining all associated hits [outlier hits are rejected]

4. selection:

sets quality flags based on the fit χ^2 and the track compatibility w/ interaction region

aims to reject fake tracks **tracks sharing too many hits are also cleaned as duplicates**



track reconstruction in CMS

in each iteration, tracks are reconstructed in four steps:

1. seeding: use combination of pixel, strip or mixed hits

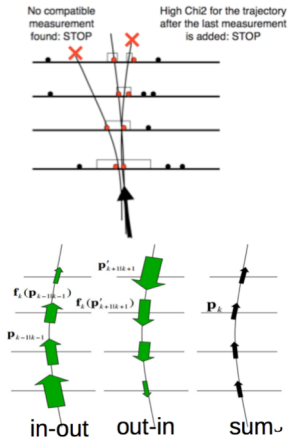
provides track candidates,

with an initial estimate of the trajectory parameters and their uncertainties

2. pattern recognition: alignment uncertainty taken into account

hit compatible w/ predicted track position are added (Kalman update)

to the trajectory track parameters are updated



3. final fit: taking into account the Field non uniformity and a detailed description of the material budget

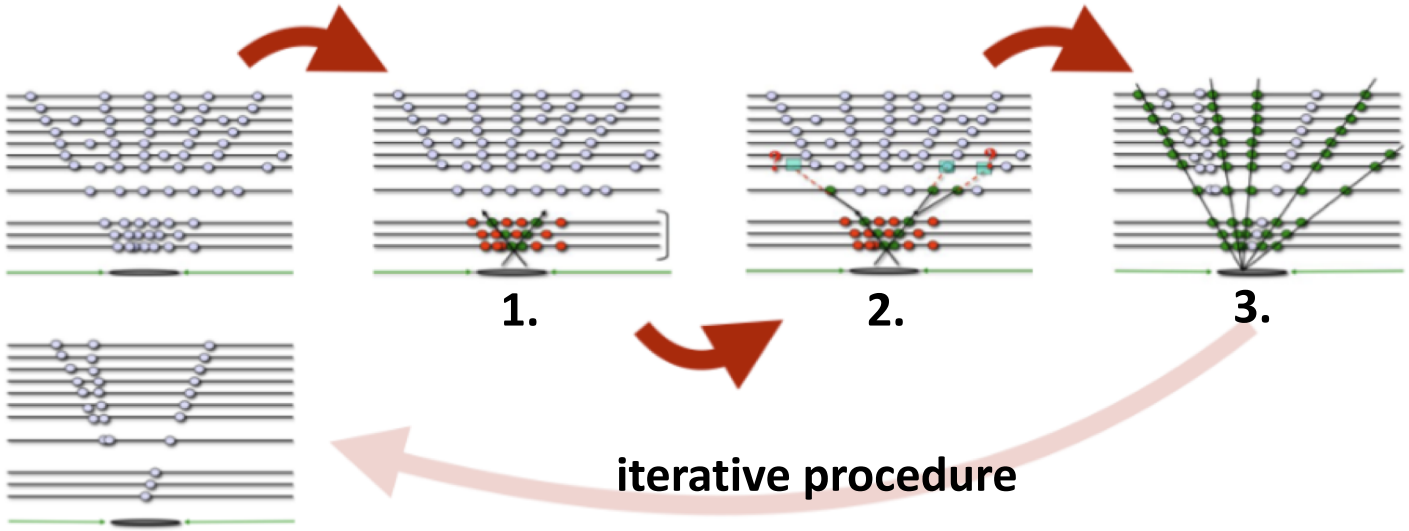
provides the best estimate of the parameters of each smooth trajectory

after combining all associated hits [outlier hits are rejected]

4. selection:

sets quality flags based on the fit χ^2 and the track compatibility w/ interaction region

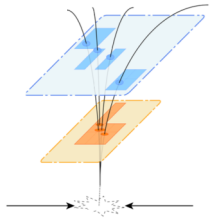
aims to reject fake tracks **tracks sharing too many hits are also cleaned as duplicates**



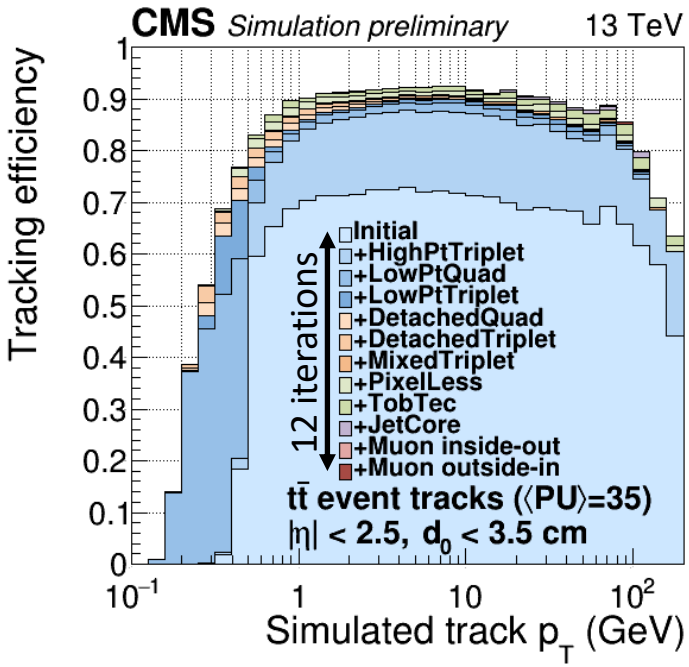
Iterative Tracking in CMS

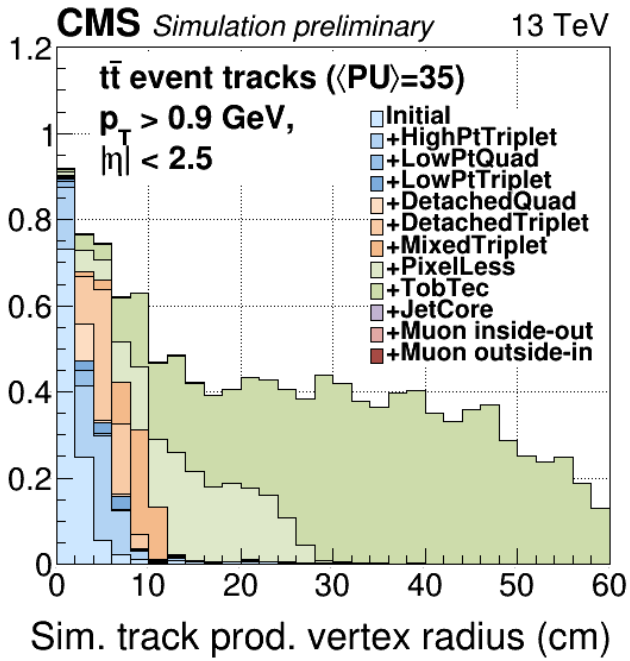
tracks reconstruction is an **iterative procedure**:

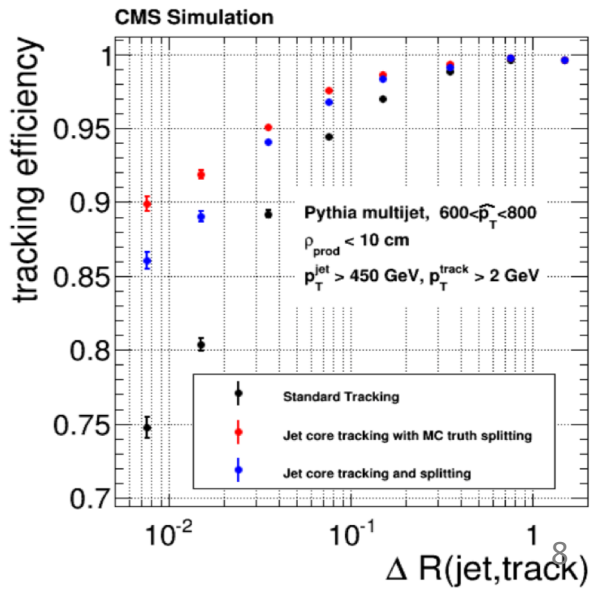
- the *InitialStep* makes use of high-p_T quadruplets coming from the beam spot region
- subsequent steps use triplets, or improve the acceptance either in p_T or in displacement
- the later steps use seeds w/ hits from the strip detector to find detached tracks,
- final steps are dedicated to special phase-space
  - highly dense environment (i.e. w/in jets)
  - clean environment (i.e. muons)



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



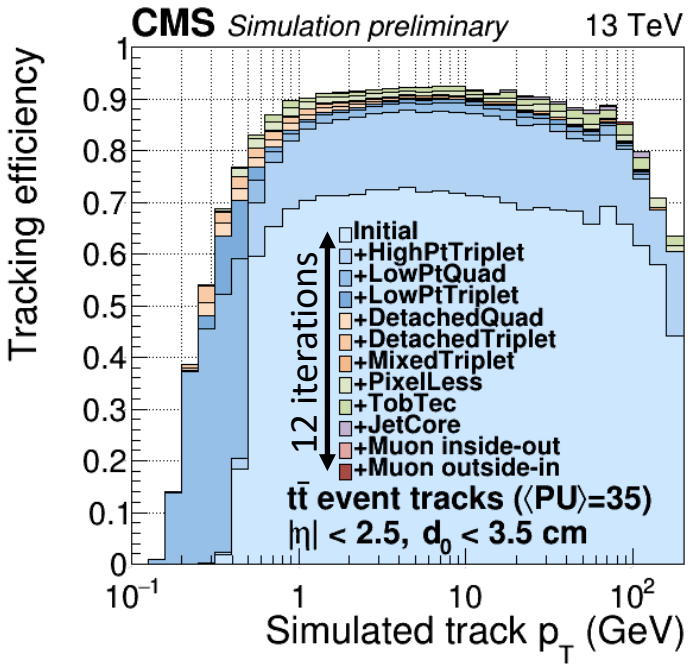


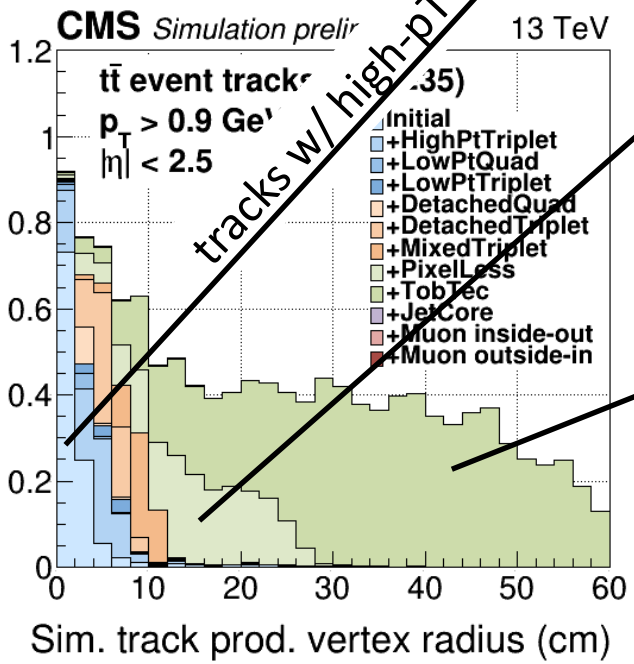


Iterative Tracking in CMS

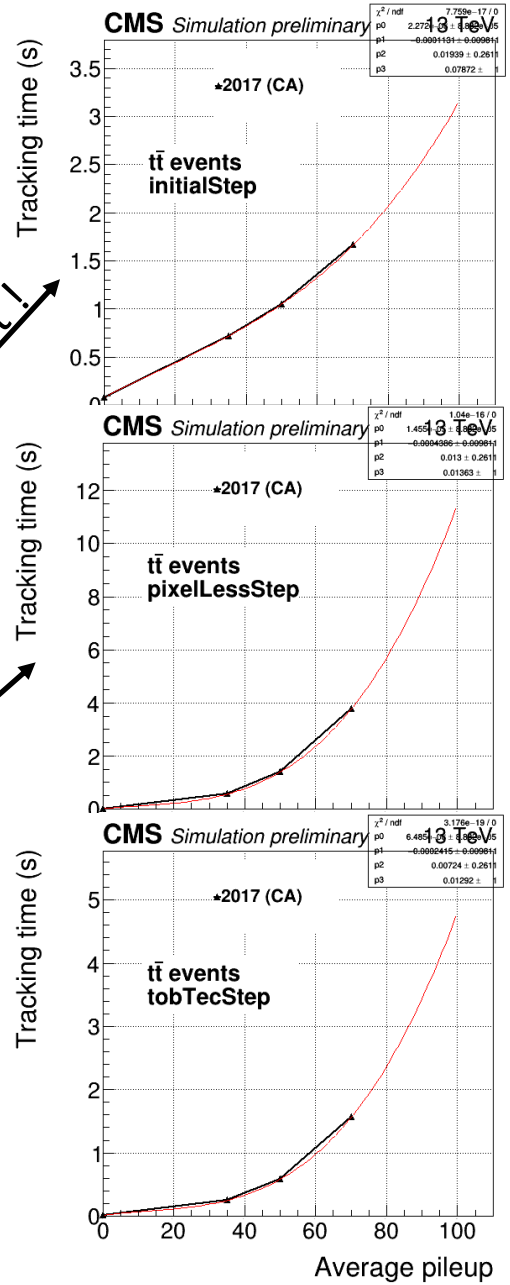
tracks reconstruction is an **iterative procedure**:

- the *InitialStep* makes use of high-p_T quadruplets
  coming from the beam spot region
- subsequent steps use triplets,
  or improve the acceptance either in p_T or in displacement
- the later steps use seeds w/ hits from the strip detector
  to find detached tracks,
- final steps are dedicated to special phase-space
  - highly dense environment (i.e. w/in jets)
  - clean environment (i.e. muons)





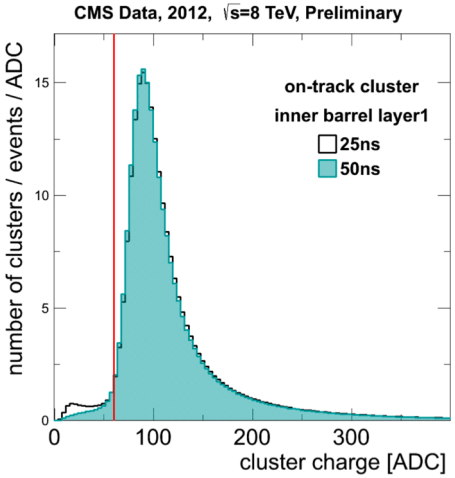
tracks w/ high-pT can be reconstructed fast !



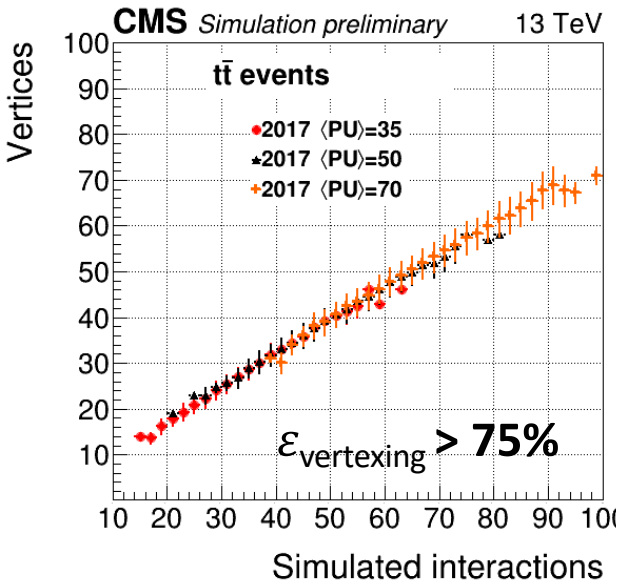
Run2 performance

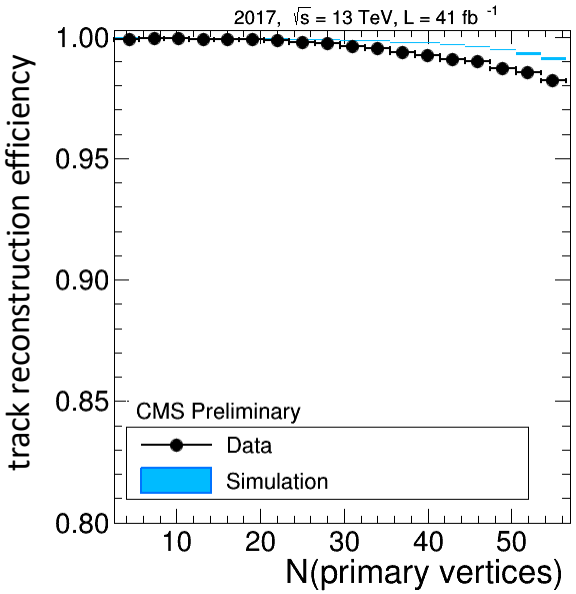
few challenges :

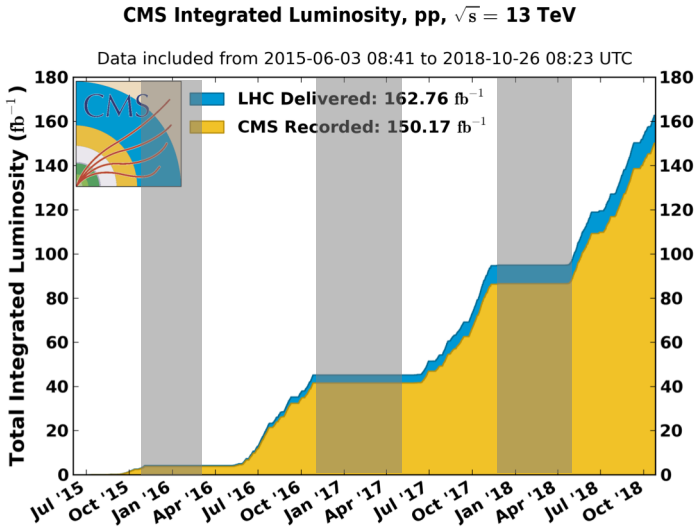
- 50ns → 25ns **out-of-time pile up**
  introduce few mitigations
  like the (strip) cluster charge cut

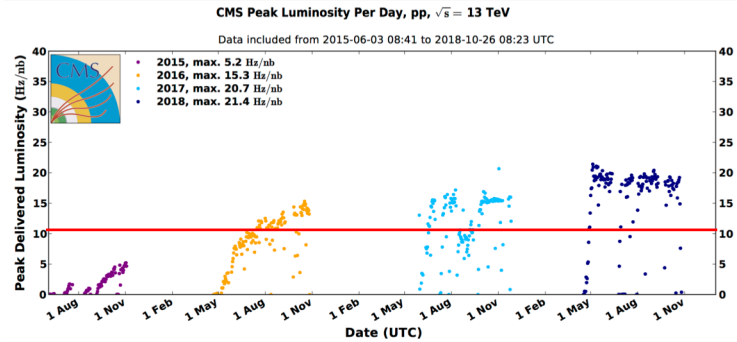


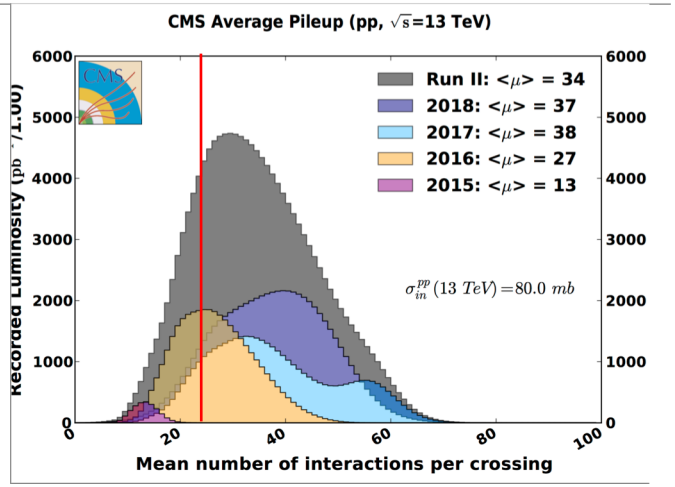
- much **higher PU** and **luminosity** (w.r.t. Run1 and design)







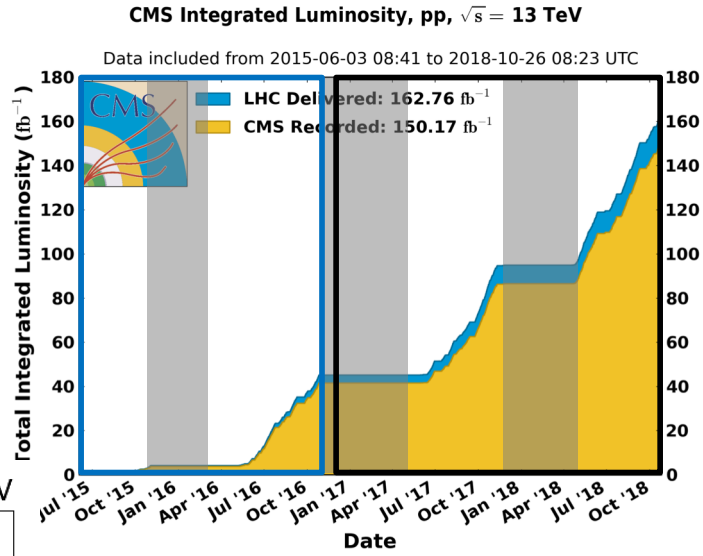


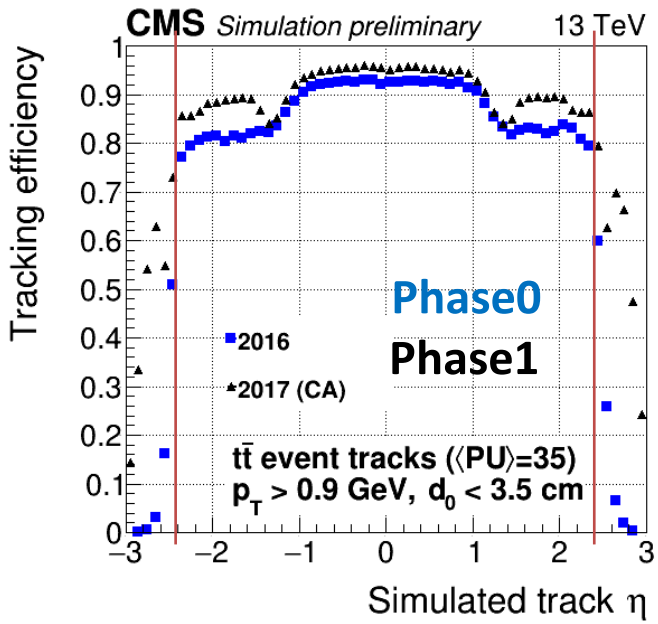


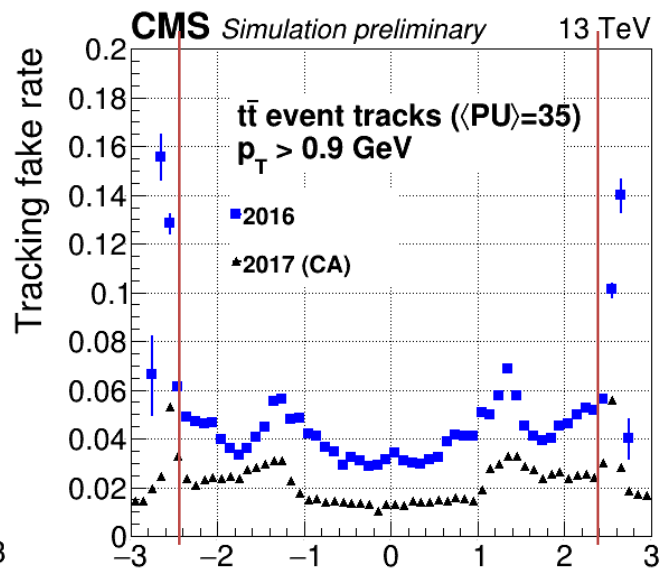
Run2 performance

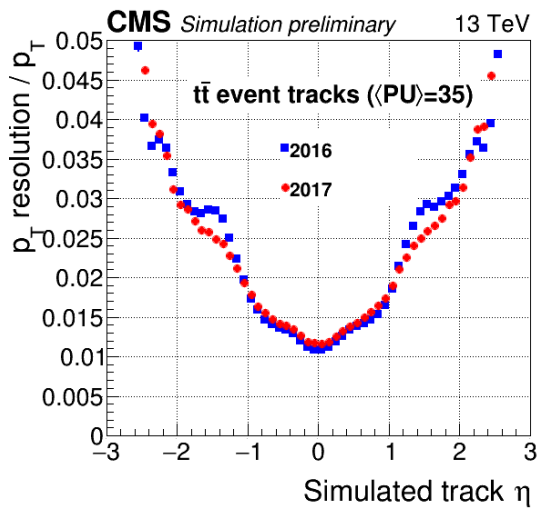
few challenges :

- 50ns → 25ns **out-of-time pile up**
- much **higher PU** and **luminosity** (w.r.t. Run1 and design)
- commissioning the **new pixel detector** during the run

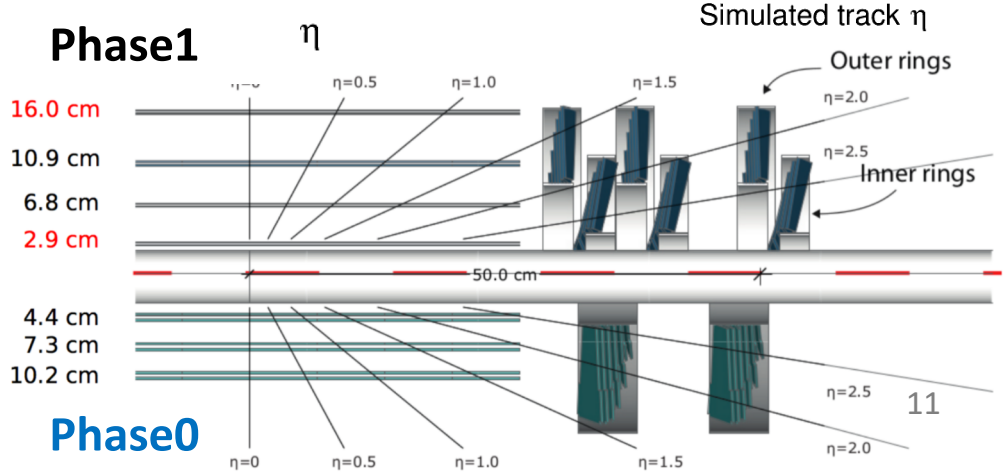








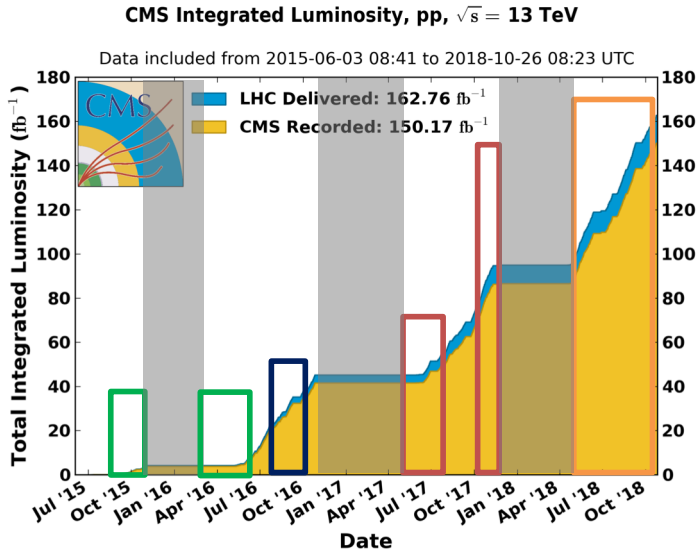
- ✓ increase efficiency
  (above all at high pseudo-rapidity)
- ✓ decrease fakerate
- ✓ improve pT resolution
  (mainly in the transition region)

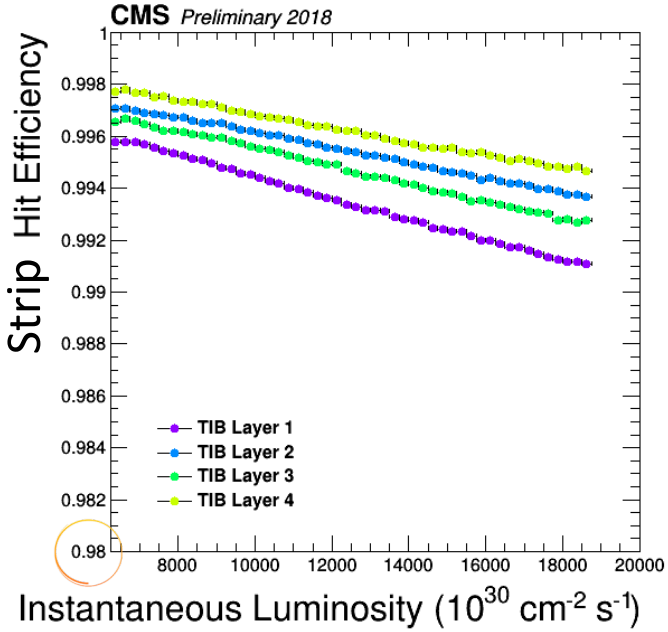


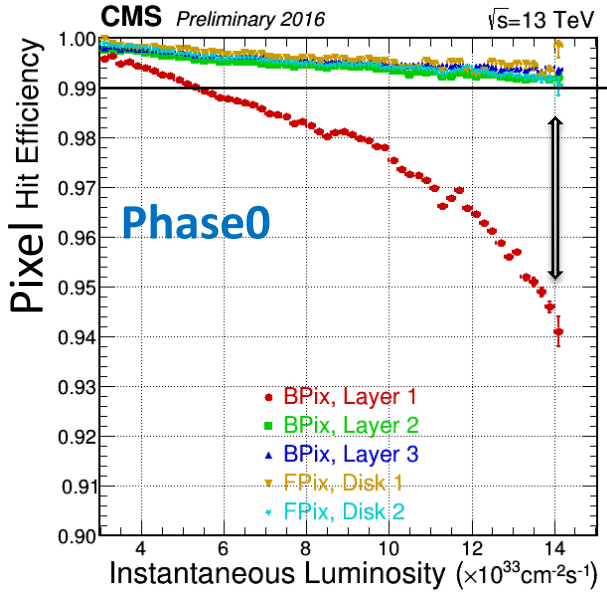
Run2 performance

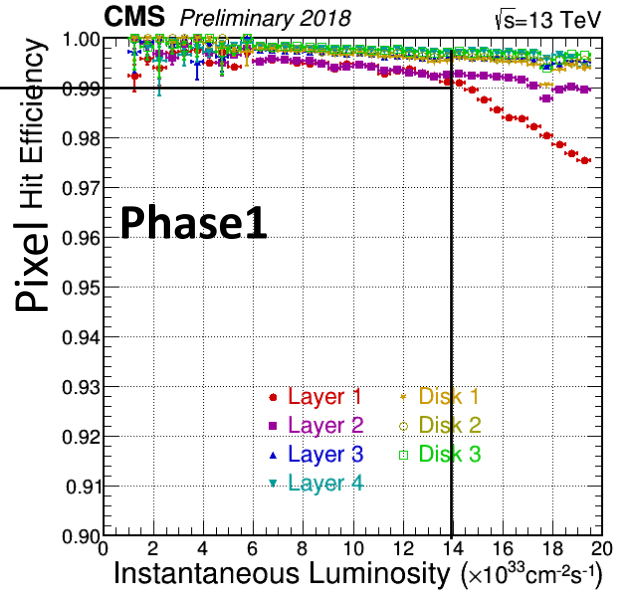
few challenges :

- 50ns → 25ns **out-of-time pile up**
- much **higher PU** and **luminosity** (w.r.t. Run1 and design)
- commissioning the **new pixel detector** during the run
- some detector issues/features









extrapolate track to tracker module and check for the presence of a reconstructed hit
[inactive/problematic modules crossed out]

single hit reconstruction efficiency : ***≥ 99%***

the inefficiency mainly depends on the particle flux, inner layers are more sensitive than the outer ones

Run2 performance

in general,
high tracking and **vertexing performance** (despite difficult circumstances),
thanks to significant improvements,
which were made during both the LS1 and Run2:

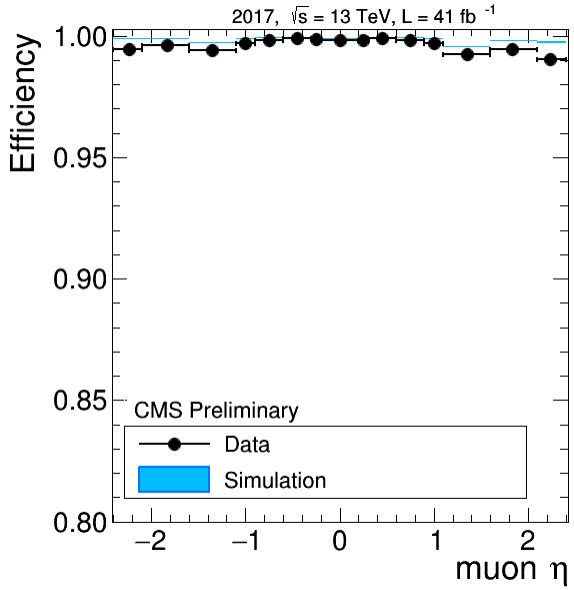
- new iterations, new tuning, PU mitigation,
- code re-engineering, new seeding framework,
- **Cellular Automaton seeding**, mitigation strategy, etc..

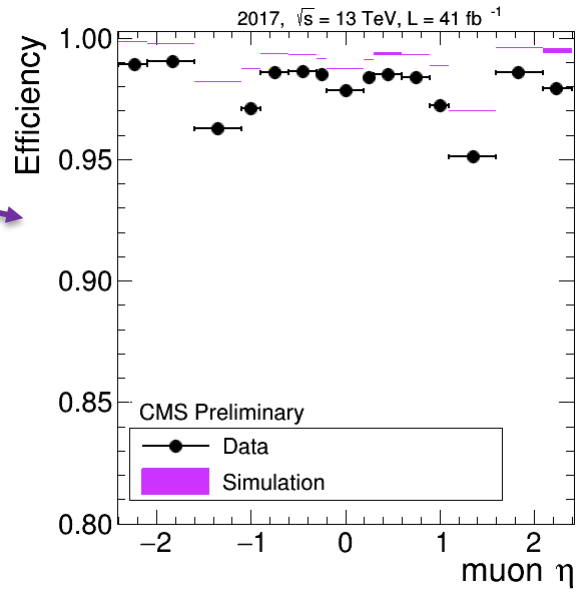
	2017	2018
tracker + muon seeded	99.0%	**99.5%**
tracker-only seeded	97.0%	**98.0%**

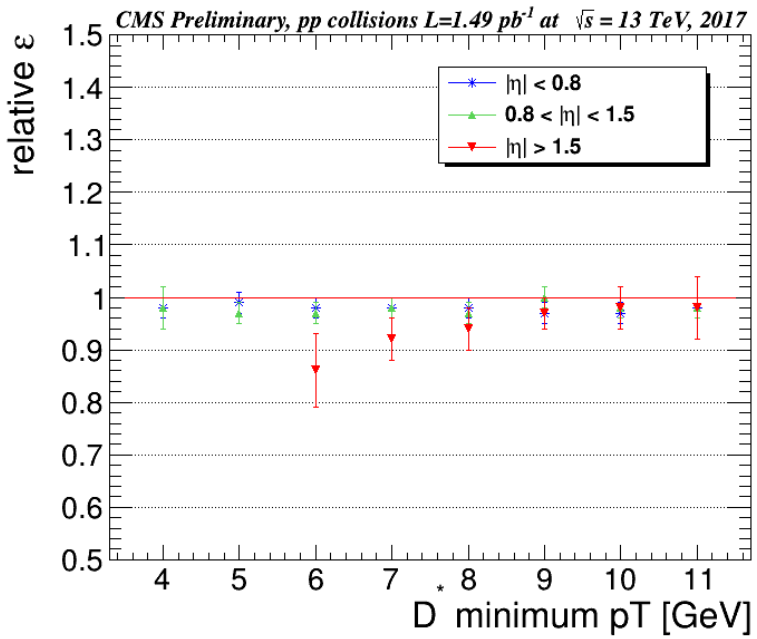
tracking efficiency using events from $Z \longrightarrow \mu\mu$ and $J/\Psi \longrightarrow \mu\mu$

$\longrightarrow \mu$

$\longrightarrow \pi, K, ..$







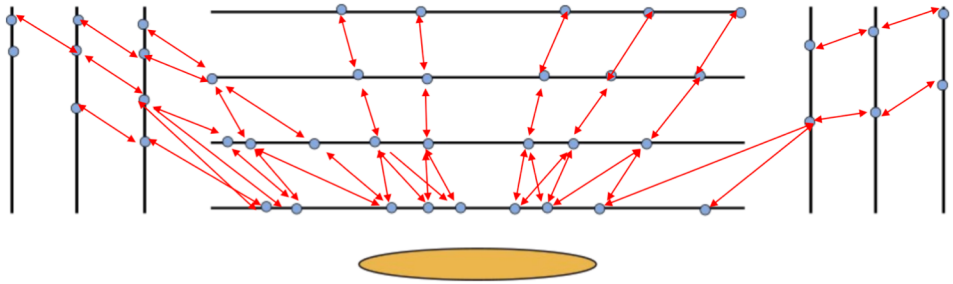
	2017	2018
$\varepsilon_{\pi,\mathrm{rel}}$	97.9%	**98.8%**

tracking efficiency using events from
$D^* \longrightarrow D^0 \pi_{\mathrm{slow}} \longrightarrow K\pi\ \pi_{\mathrm{slow}}$ and $D^* \longrightarrow D^0 \pi_{\mathrm{slow}} \longrightarrow K\pi\pi\pi\ \pi_{\mathrm{slow}}$

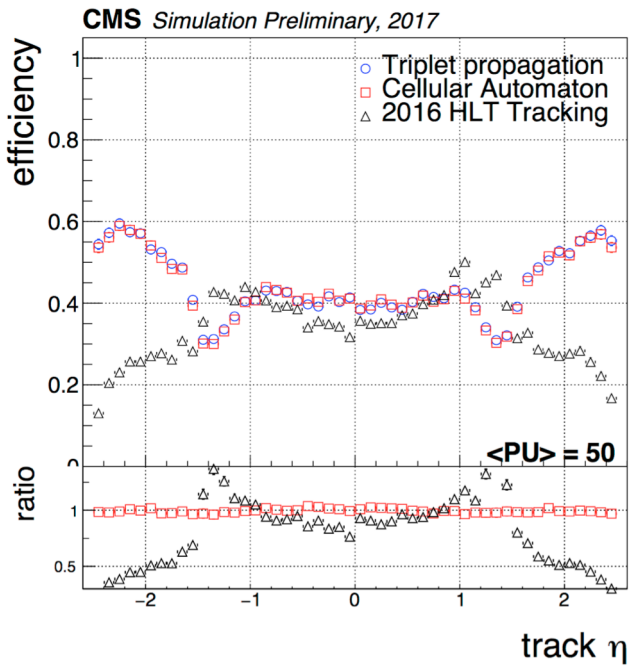
cellular automaton

new track seeding algorithm based on **Cellular Automaton** (CA) technique

- it starts from a list of layers and their pairings
  - a graph of all possible connections between layers is created
  - doublets (cells) are created for each pair of layers [compatible with a region hypothesis]
- fast computation of the compatibility between 2 connected cells

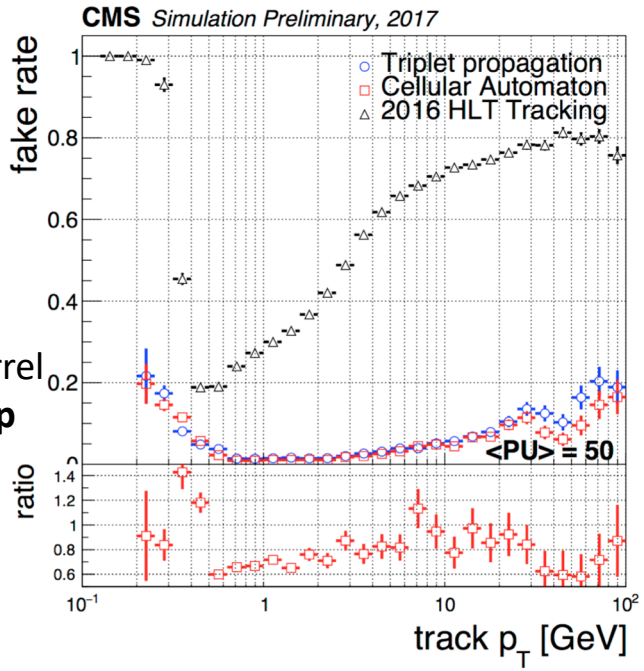


Quadruplet Algorithm	Time per event	speedup wrt. 2016
GPU Cellular Automaton	(1.2 ± 0.9) ms	24.4×
CPU Cellular Automaton	(14.0 ± 6.2) ms	2.1×
Triplet Propagation	(72.1 ± 25.7) ms	0.4×
2016 Pixel Tracks	(29.3 ± 13.1) ms	1×

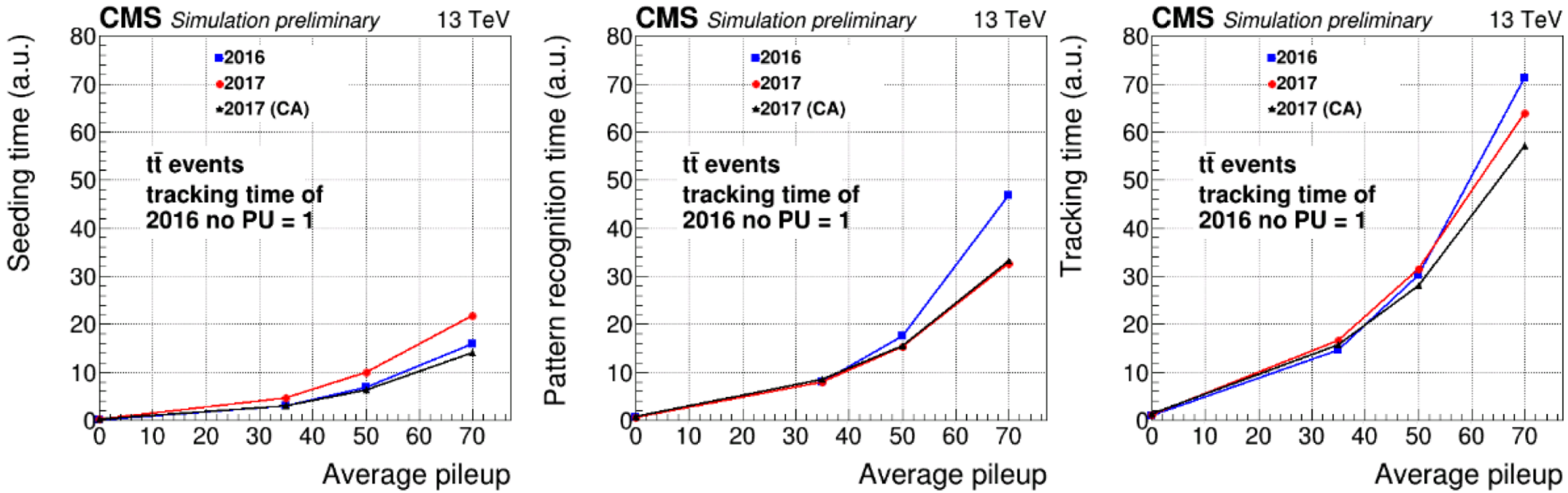


✓ timing
- **x5 faster** than old algorithm
  x2 faster than 2016 configuration

✓ performance
(with new pixel detector)
- almost same efficiency in the barrel
- **50% efficiency gain in the endcap**
- **x4 reduction in fakes** !



cellular automaton



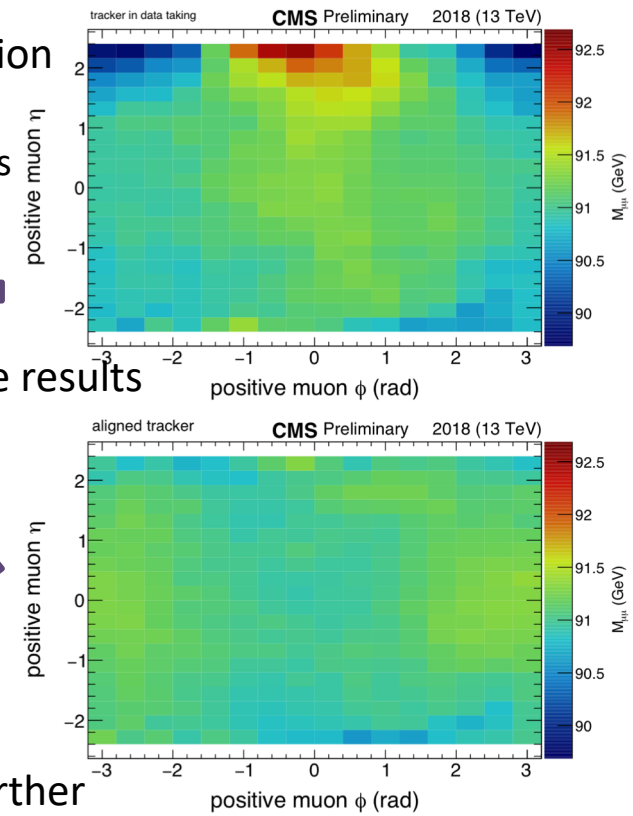
cellular automaton seeding

- more robust
- smaller complexity vs PU than 2016 track seeding
  despite the increased number of layer combinations involved
- in pattern recognition, no additional gain

despite the increase in the number of pixel layers
~20% faster track reconstruction wrt to 2016 tracking @ <PU> = 70

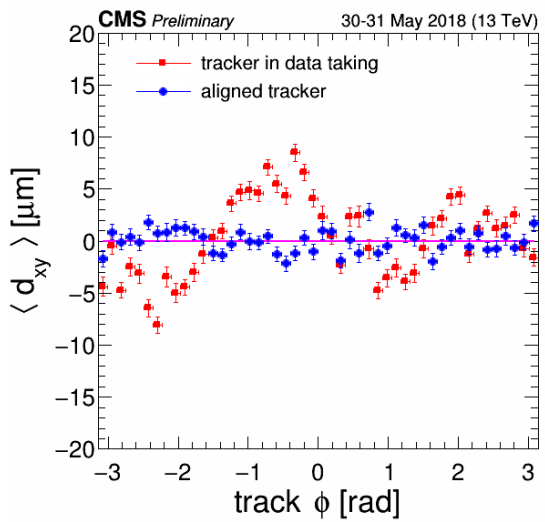
Run2 performance

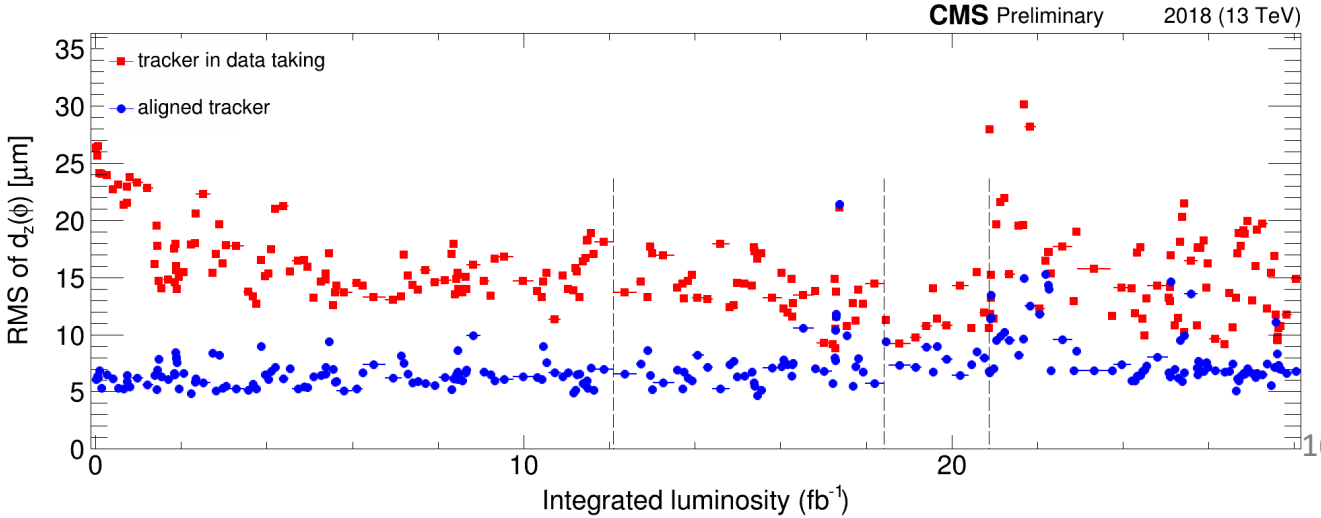
- **precise** knowledge of the **detector alignment** crucial for track reconstruction
- detector components have been observed to move
  e.g. tension of support structure due to thermal transients or magnetic field changes
- CMS employs an **automatic procedure**
  to monitor movements of top level hierarchical mechanical structures (half-barrels and half-disks)
  when appropriate, detector geometry is updated based on these online results



- ➢ ϕ-dependent **mass bias in $m_{\mu\mu}$ @high muon rapidity**
  → greatly reduced w/ the update of the alignment
- ➢ track impact parameters (d_{xy} and d_z) are sensitive to
  - Lorentz Angle mis-calibrations
  - misalignment in the pixel detector

  → residual bias is nicely recovered by the time granularity alignment
  → still sub-optimal performance in certain period could be improved further





track reconstruction @HLT

in Run2, we decide to keep the Tracking@**HLT as close as possible** to the **offline** track reconstruction on both the algorithms & configurations but, **time** is a constraint

⇒ **reduce #iterations** wrt offline tracking

⇒ **constrain** tracking regions (i.e., **regional** tracking)

CA seeding since 2017

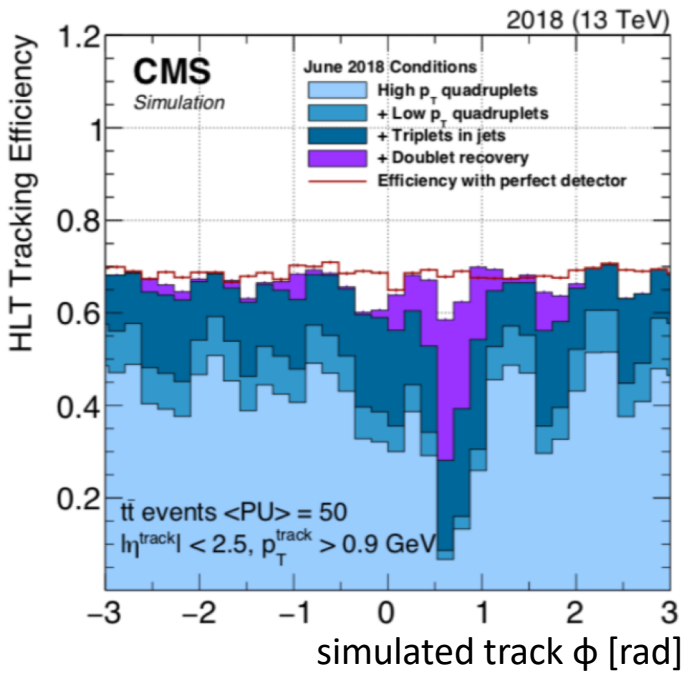
N	Step Name	Seeding	Target Track
0	Iter0	pixel tracks (from **quadruplets**)	prompt, high p_T
1	Iter1	pixel quadruplets	prompt, low p_T
2	Iter2	pixel triplets	recovery (not only high p_T)
3	Triplet recovery	pixel triplets in η-ϕ region	static triplet recovery
4	Doublet recovery	Pixel doublets in η-ϕ region	static doublet recovery

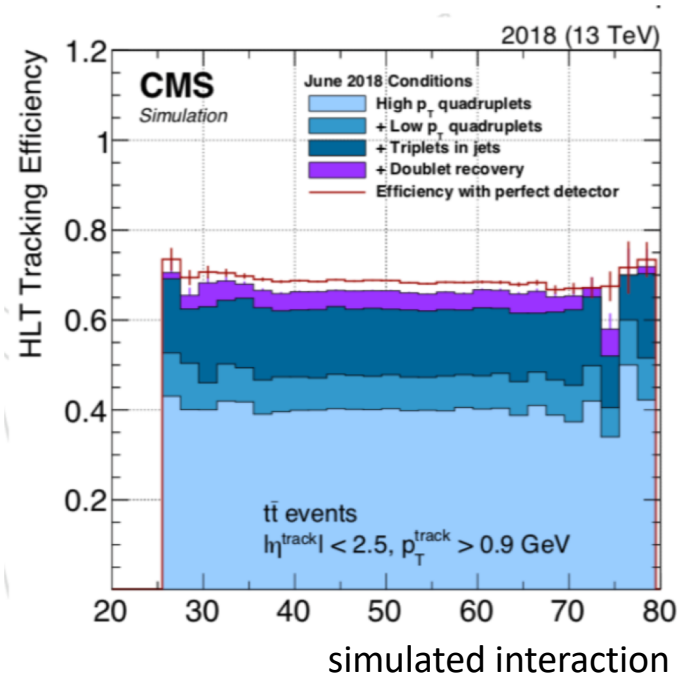
during Phase1 pixel commisioning,
failures observed mostly geometrically contained

⇒ in 2017, adopted a **Static mitigation** via dedicated iterations in specific *η-φ* regions

however, recovery is insufficient for additional (dynamic) pixel issues [like the DC/DC converter issue]

⇒ in 2018, adopted the **Dynamic mitigation** of pixel issues [trade off among efficiency/fake and timing]





nearly ideal efficiency is achieved

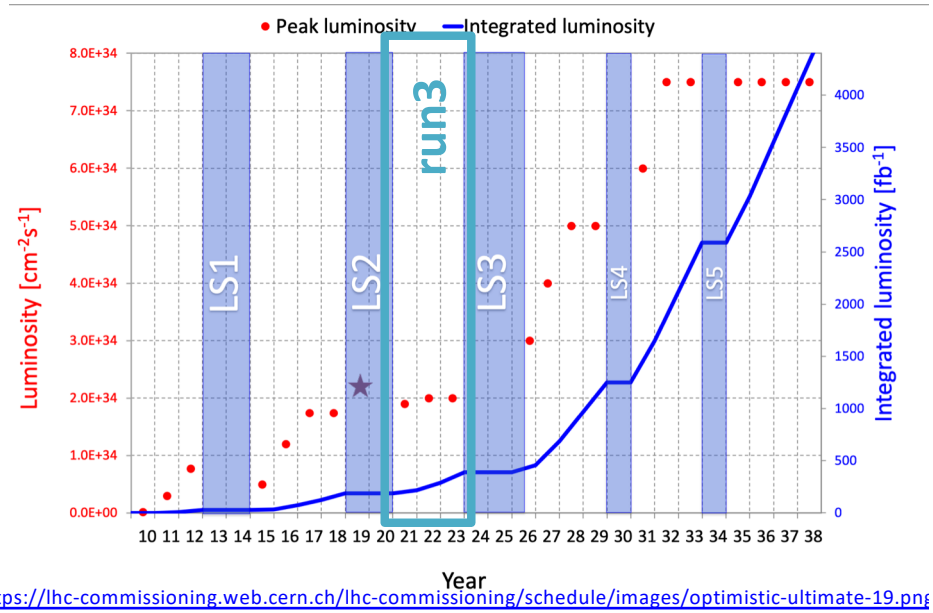
efficiency is almost flat as a function of #PU

plans : run3

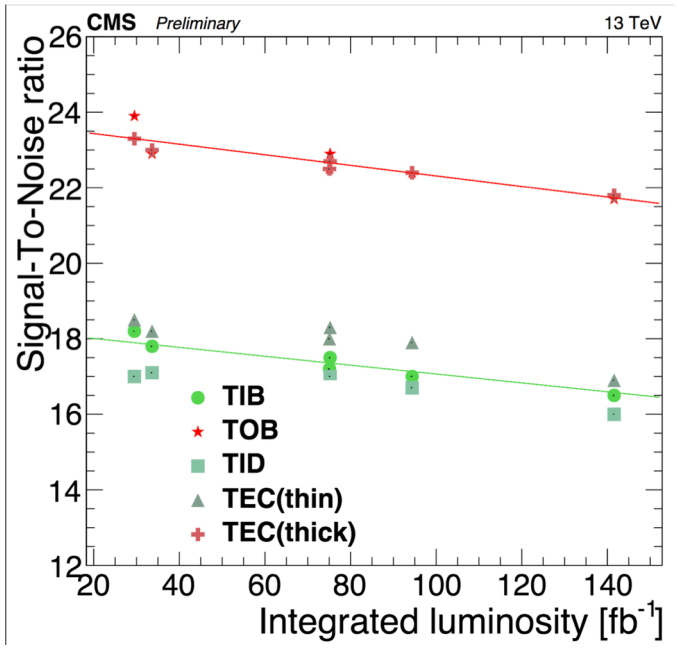
during Run3,
there will be a not negligible degradation of both Pixel and Strip detectors wrt nominal performance due to the accumulated radiation

- large portions of TIB1/2 inactive due to leakage currents

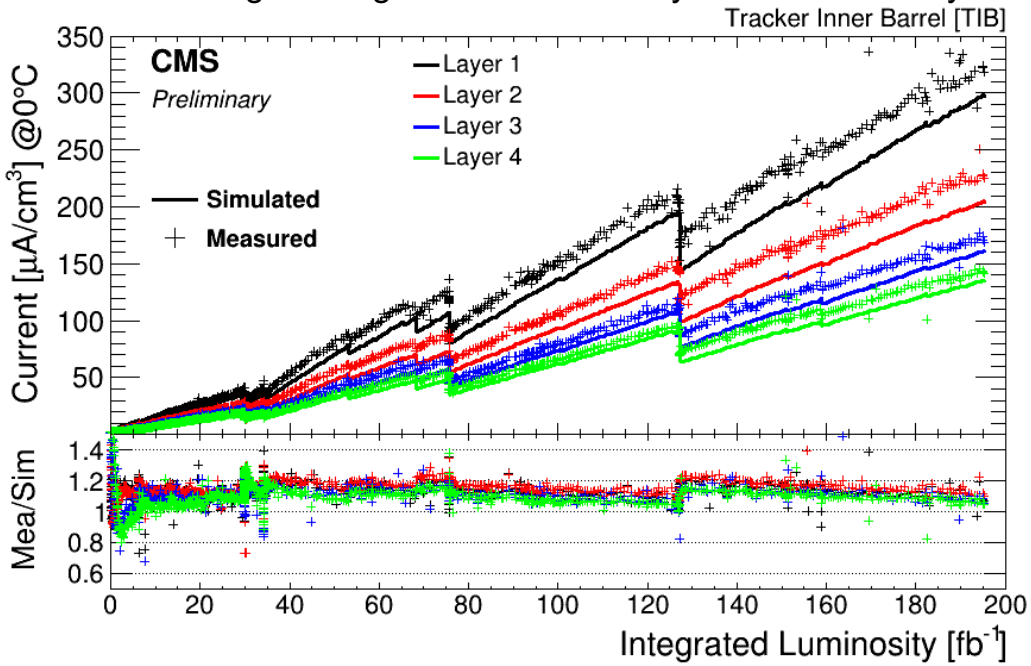
⇒ **mitigation strategies need to be developed**



https://lhc-commissioning.web.cern.ch/lhc-commissioning/schedule/images/optimistic-ultimate-19.png

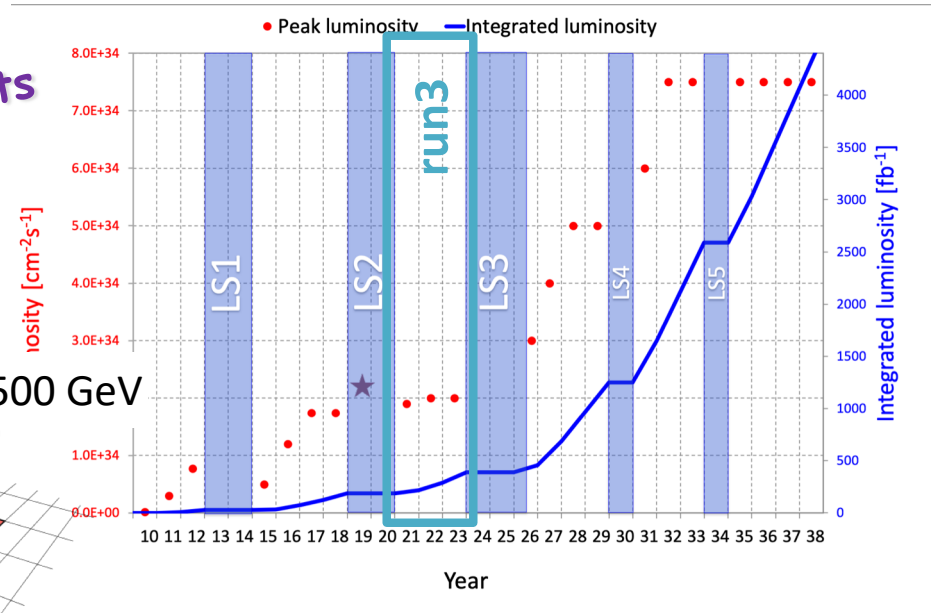


Average leakage current for TIB layers with luminosity

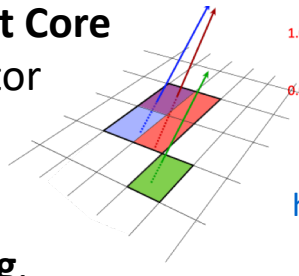


plans : run3

DeepCore
tracking inside jets



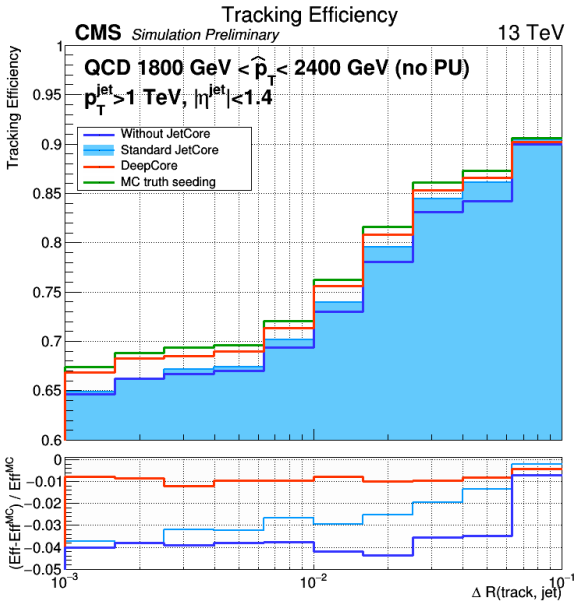
- algorithms like the Particle Flow, the b-tagging and the jet sub-structure rely on the good performance of the track reconstruction w/in jets
- tracking inside jets starts to become inefficient for $p_T > 500$ GeV
- in Run2, dedicated step has been added : **Jet Core**
  → the merged clusters in the pixel detector affect already the seeding step

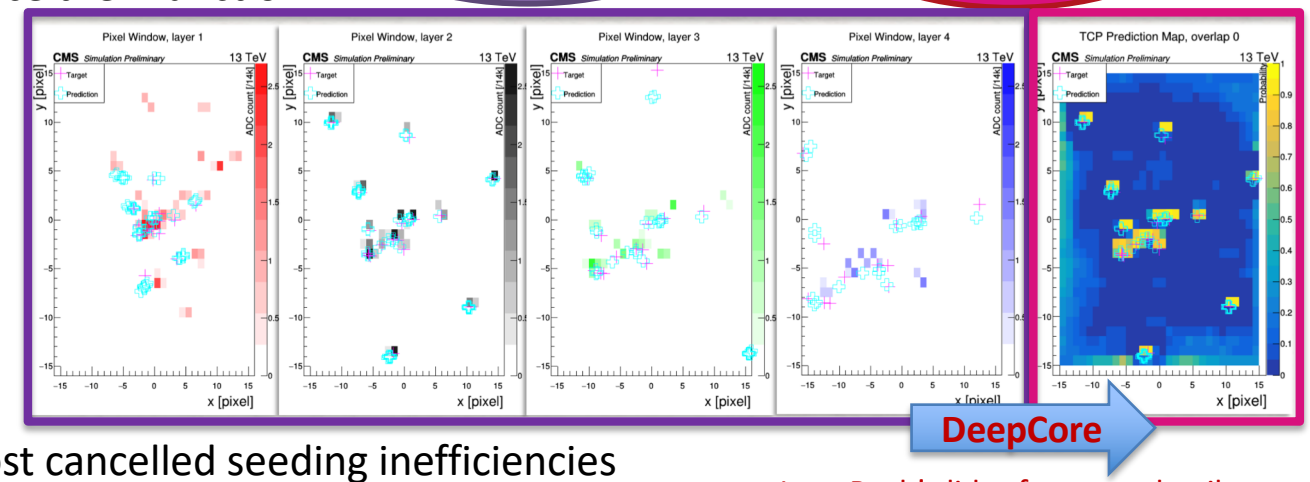


hits from different tracks can result in a merged pixel cluster in the core of high-p_T jets
[the effect is even more pronounced for b-jets, because the B decay happens closer to the pixel detector]

- in Run3, the **DeepCore** will be deployed
  ⇒ basic idea is to **skip the pixel clustering**, **exploit directly the RAW pixel data** and **reconstruct the seed of tracks w/in the jets**
  → develop a convolutional Neural Network (cNN) to reproduce the «function»

4-layers pixel maps → DeepCore → tracks seeds





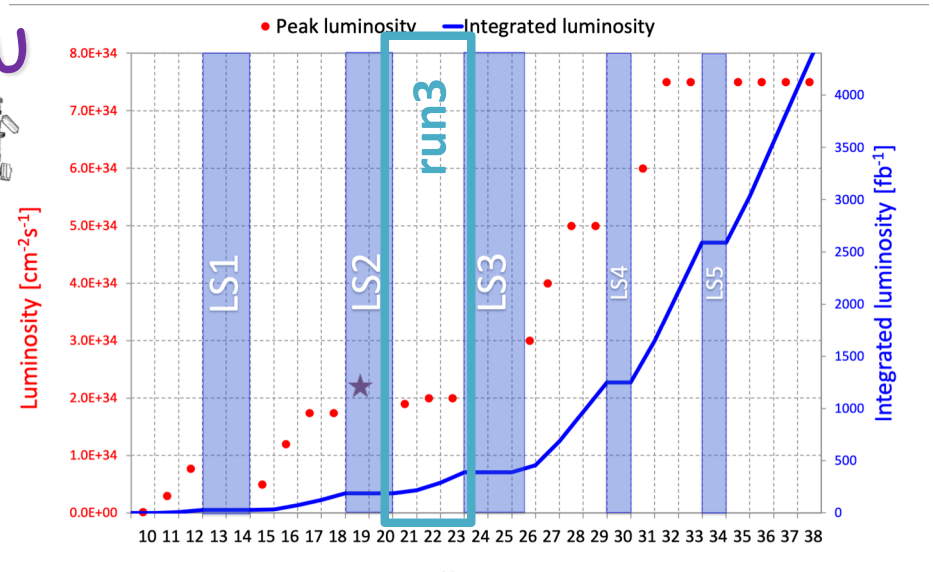
DeepCore

- ✓almost cancelled seeding inefficiencies
- ✓**fake rate reduction up to 60%**
- ✓**seeding timing reduced by 85%**

see Jean-Roch' slides for more details on ML approaches in CMS (and Tobias Golling's in ATLAS)

plans : run3 tracking on GPU

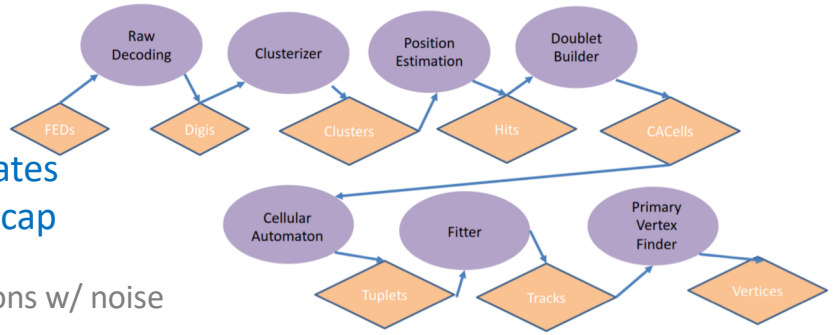
Patatrack

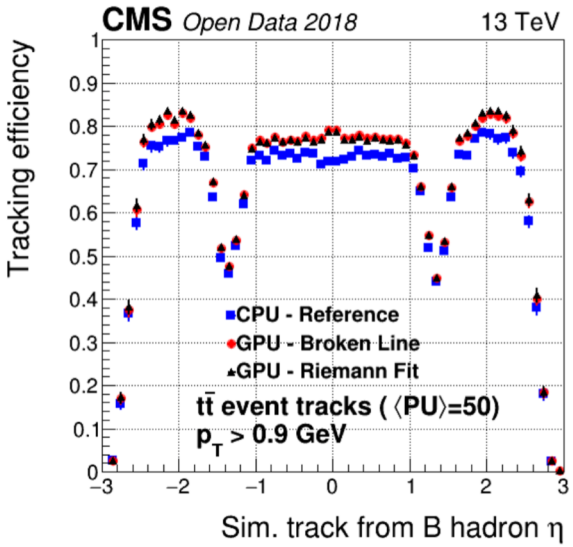


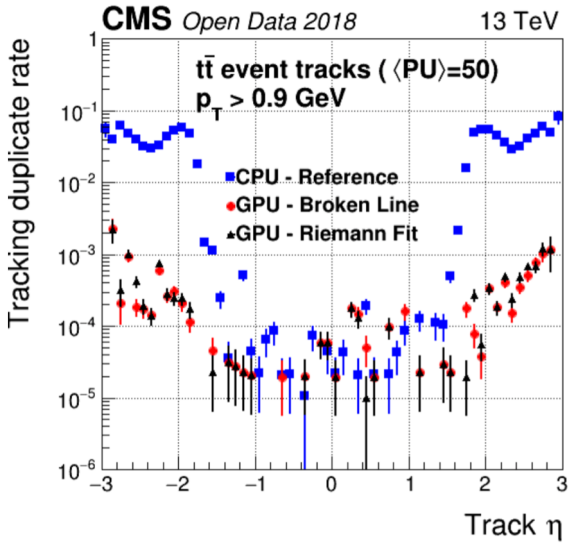
- to ensure smooth operation of the **heterogeneous HLT farm** during **Run4**
- accelerated RAW data to Pixel Track and Vertices reconstruction
- currently, at PU = 50, pixel tracks are reconstructed only for ~10% of the events @HLT
- profit from the upgrade of the Pixel to **redesign the pixel track algorithm**
  - developed 2 **Multiple Scattering-aware fits**:
    - Riemann Fit
    - Broken Line Fit

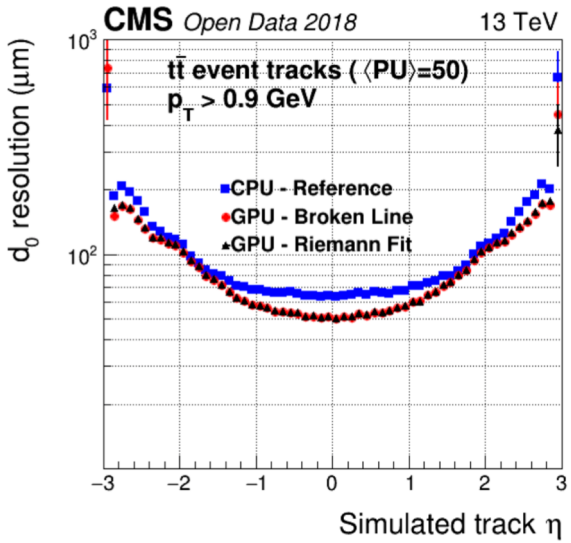
    improve parameter resolutions and fake rejection
  - developed a **seeds cleaner** "fishbone" mitigate duplicates in the pixel endcap
  - implemented the **z-clustering by DBSCAN**

    density-based spatial clustering of applications w/ noise



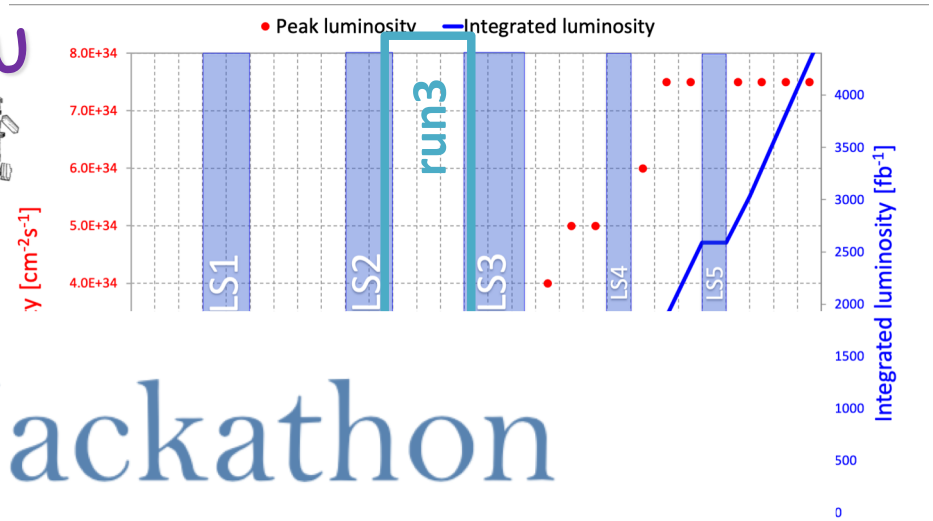






plans : run3 tracking on GPU

Patatrack



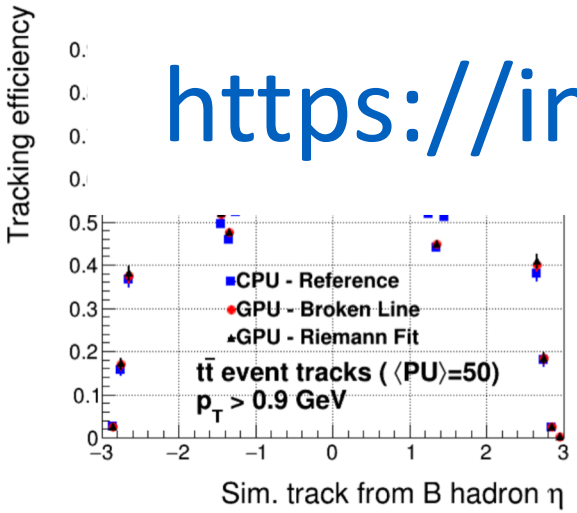
- to ensure smooth operation of the **heterogeneous HLT farm** during **Run4**
- ac to
- cu or
- pr

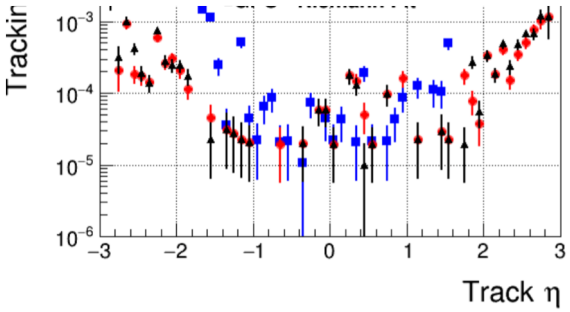
6th Patatrack Hackathon

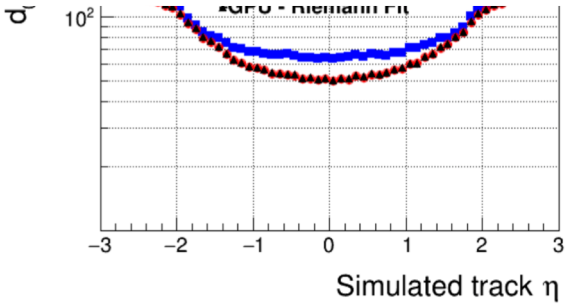
Save the date!

CERN Idea Square, 1-5 July 2019

https://indico.cern.ch/event/799486

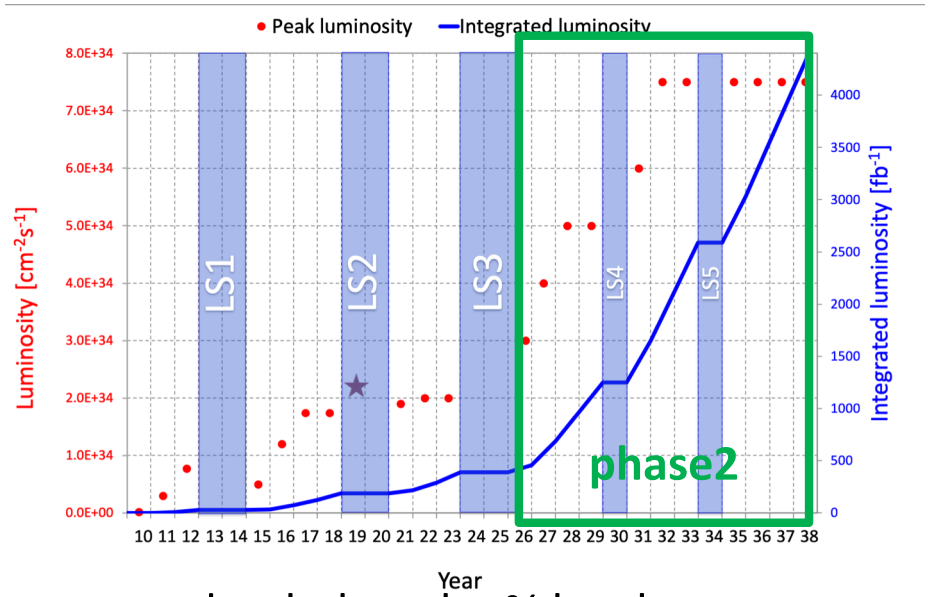






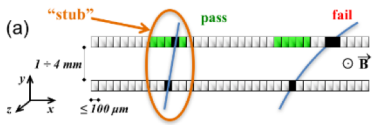
plans : phase2

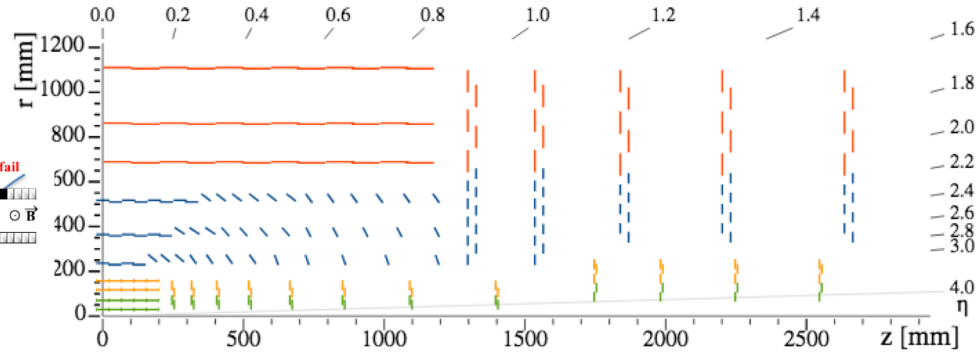
- present tracker designed for
  an integrated lumi of 500/fb and <PU> ~ 30-50

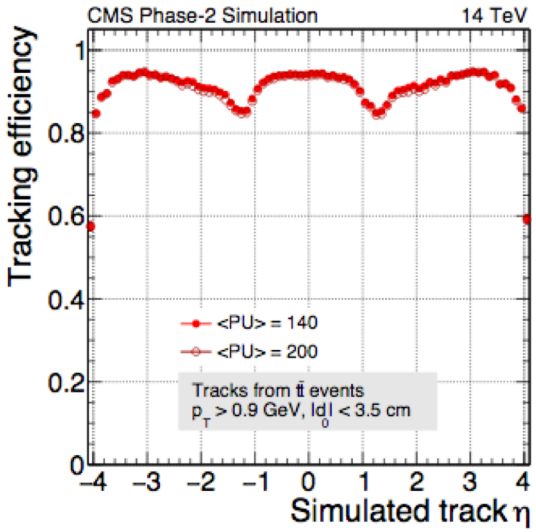


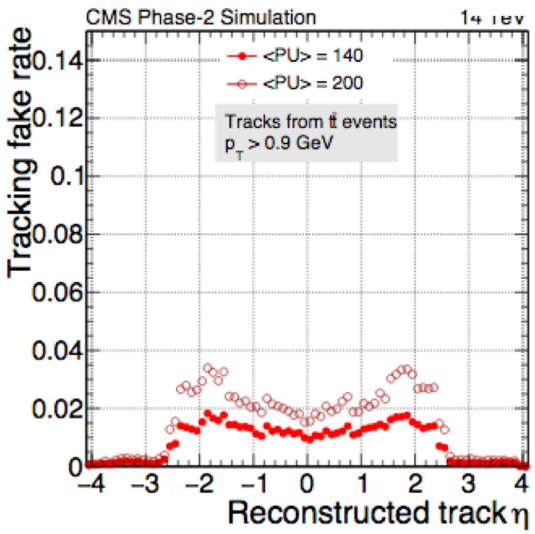
Requirements for Phase2

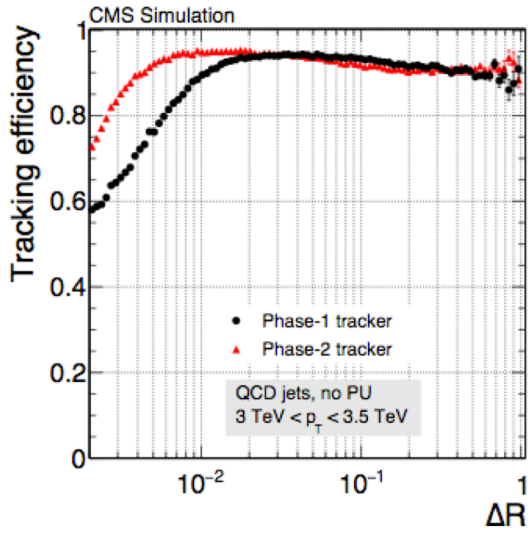
- High radiation tolerance to operate efficiently
  up to 3000/fb
- increased granularity to maintain channel occupancy around or below the % level
- reduced material in the tracking volume
- contribution to the Level-1 trigger
- extended tracking acceptance
- robust pattern recognition











Step Name	Seeding	Target Tracks
HighPtQuad	pixel quadruplets	prompt, high p_T
HighPtTriplet	pixel triplets	prompt, high p_T, recovery
LowPtQuad	pixel quadruplets	prompt, low p_T
LowPtTriplet	pixel triplets	prompt, low p_T, recovery
DetachedQuad	pixel quadruplets	displaced
PixelPair	pixel pairs	high p_T, recovery
Muon Inside-Out	muon-tagged tracks	muon
Muon Outside-In	muon-tagged tracks	muon

see Daniel' slides for more details on Flavour Tagging at HL-LHC

conclusion

- despite challenging circumstances at the LHC in Run2
  high tracking and **vertexing performance**
  → large performance improvements
  due to the **upgrades to the pixel**
- the simulation is accurate on predicting tracking behaviour
- detailed studies of detector **material**
- track **efficiency** measurements based on data-driven techniques
- **alignment** has become an even more challenging task
  as the experiments cope w/
  the impact of detector movements and increased precision

- **new developments** targeting **Run3**
  mitigation strategy, tracking in dense environment,
  pixelTracks, and ML oriented:
  DNN track selection, DeepCore, see Jean-Roch Vlimant's talk
  CNN mergeCluster, CNN cluster shape filter

key-words:
- ❑ **pileup dependence mitigation**
- ❑ **accelerators**
- ❑ **machine learning**
- ❑ **take advantage of upgraded systems ;)**

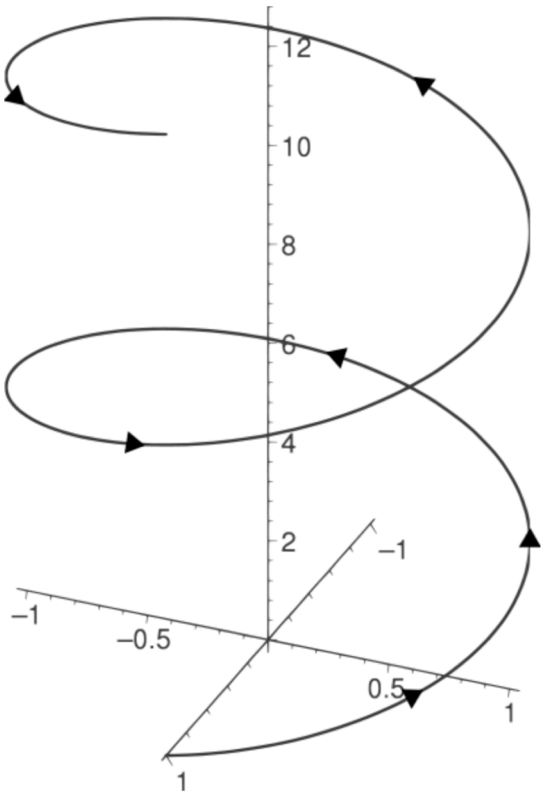
- **synergy** w/ the **tracker experts group**
  and **the physics object reconstruction experts**
  is a key ingredient for the developments
  and establishment of the expertise

try to not loose the expertise built in the last years,
and build a new team of experts

…paper on performance of tracking in Run2 on going

BACKUP

Trajectory parametrization



A helical trajectory can be expressed by 5 parameters,
but the parameterization is not unique.

Given one parameterization, we can always re-express the same trajectory in another parameterization.

In general terms, the five parameters are:

- **signed radius of curvature** (units of cm),
  which is proportional to particle charge
  divided by the **transverse momentum**, $\mathbf{p_T}$, (units of GeV);
- **angle of the trajectory** at a given point on the helix,
  in the plane **transverse to the beamline** (usually called $\boldsymbol{\varphi}$);
- **angle of the trajectory** at a given point on the helix
  with respect to the beamline (θ, or equivalently $\lambda = \pi/2 - \theta$),
  which is usually expressed
  in terms of pseudorapidity $\boldsymbol{\eta} = -\ln(\tan(\theta/2))$);
- offset or "**impact parameter**" relative to some reference point
  (usually the beamspot or a selected primary vertex),
  in the plane **transverse to the beamline** (usually called $\mathbf{d_{xy}} = \mathbf{d_0}$)
- **impact parameter** relative to a reference point
  (beamspot or a selected primary vertex),
  along the beamline (usually called $\mathbf{d_z}$)

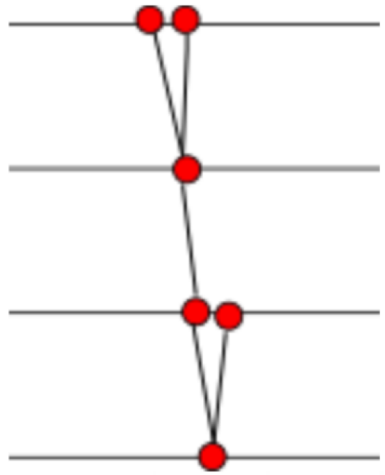
new developments: tracking on GPU

Felice Pantaleo, A. Bocci, M. Kortelainen
V. Innocente, M.Rovere

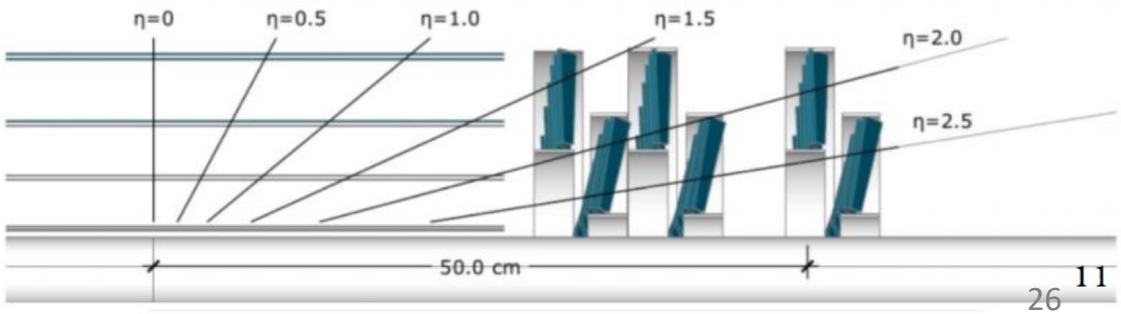
Fishbone

After using the CA for producing N-tuplets, "fishbone" seeds can be produced to account for module/layer overlaps

Only the best chi2 or highest grade n-tuplet is fitted (pentuplets in mixed barrel-endcap region)

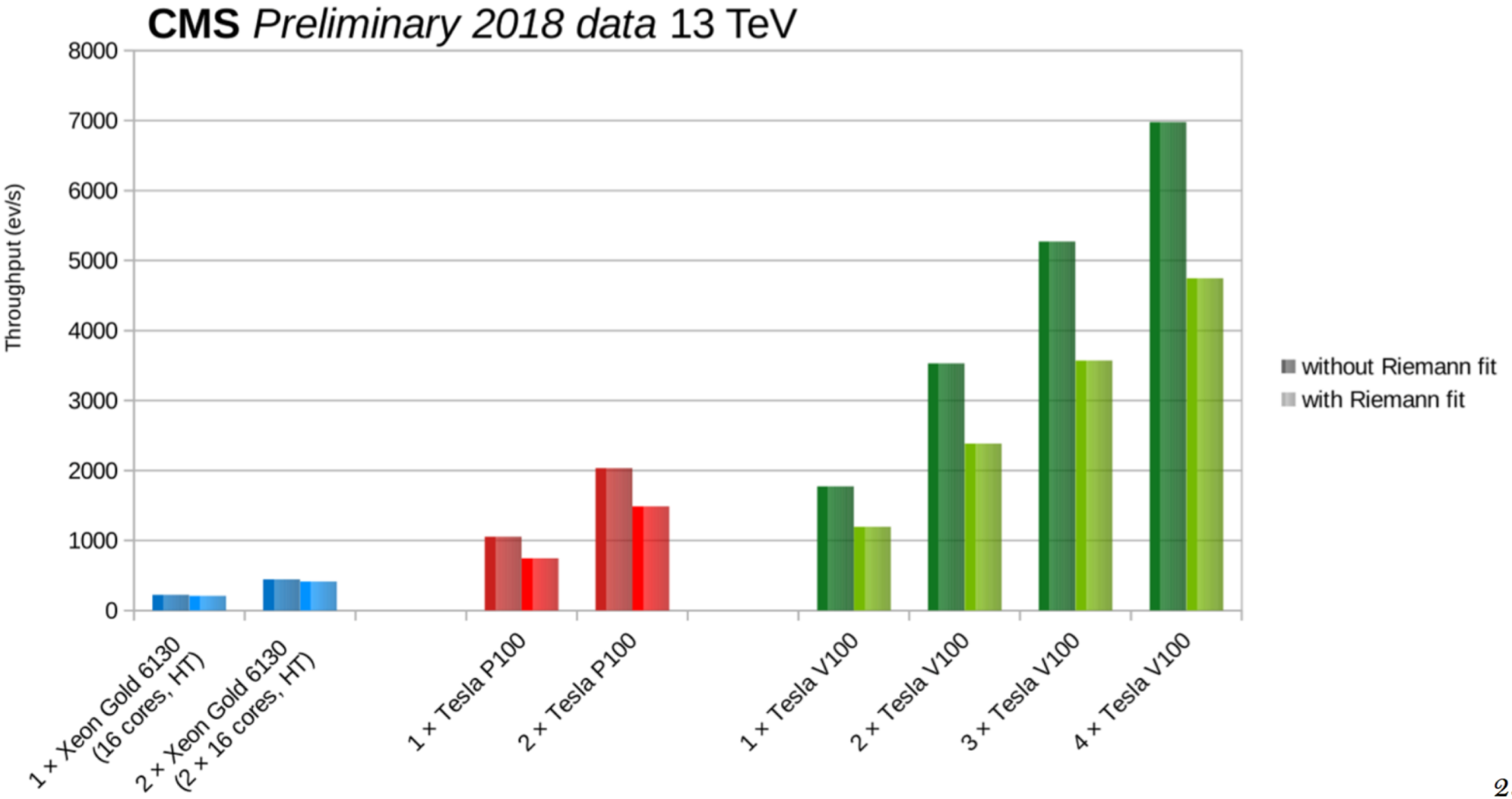


Presented for the first time as a possible mitigation of duplicates in the pixel EC in January 2017, today available in CUDA soon in C++



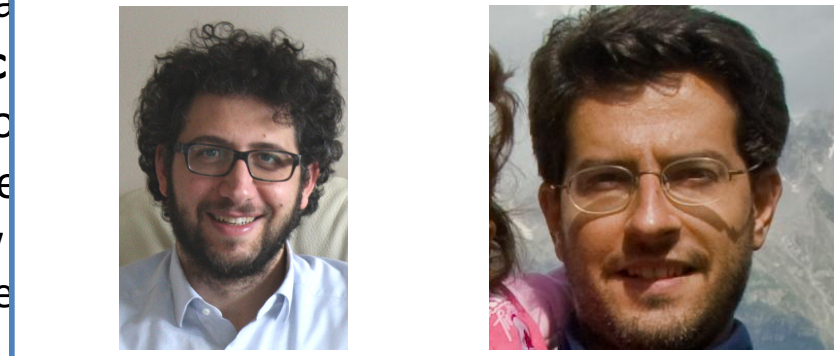
new developments: tracking on GPU

- to ensure smooth operation of the heterogeneous HLT farm during Run4
- accelerated RAW data to Pixel Track and Vertices reconstruction



Run2 performance

in genera
high trac ficult circumstances),
thanks to
which we
new
code
Cellu ..



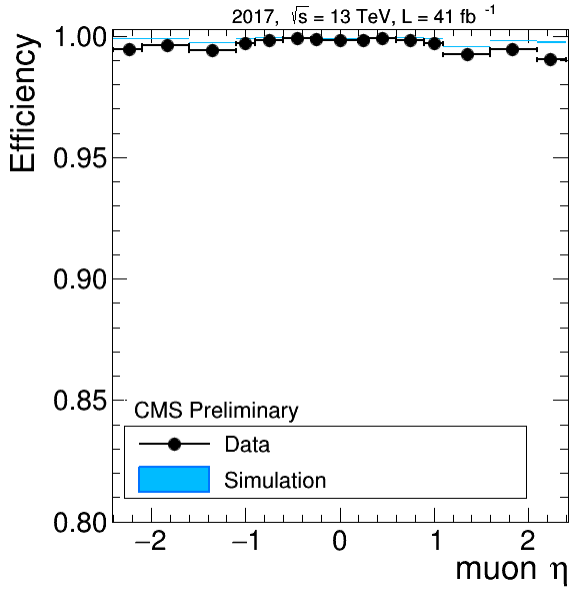
giuseppe cerati

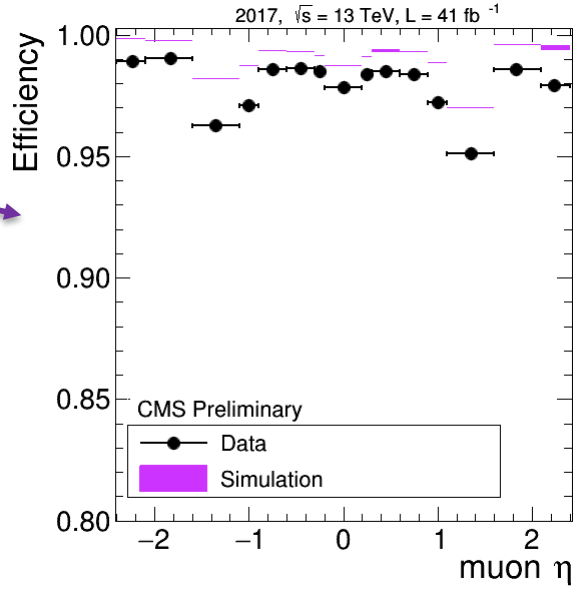
marco rovere

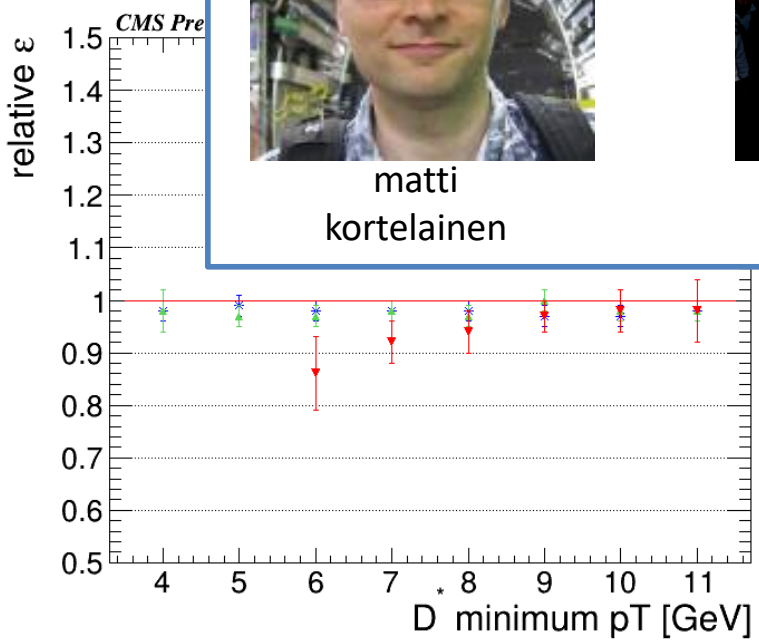
thanks to

matti kortelainen

vincenzo innocente







	2017	2018
$\varepsilon_{\pi,\mathrm{rel}}$	97.9%	**98.8%**

tracking efficiency using events from
$D^* \longrightarrow D^0 \pi_{slow} \longrightarrow K\pi\ \pi_{slow}$ and $D^* \longrightarrow D^0 \pi_{slow} \longrightarrow K\pi\pi\pi\ \pi_{slow}$

vertex reconstruction

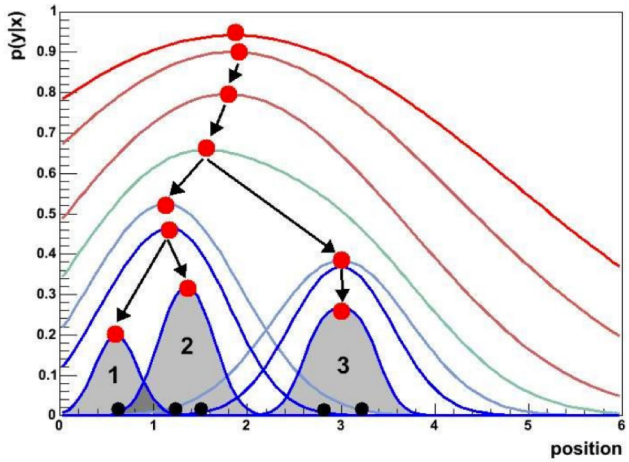
the vertex reconstruction : **Deterministic Annealing** Clustering

- Minimize χ2-like energy
- Inspired by physical annealing
  - A system reaches the state of minimal energy by going from high T to low T
  - Treat it as an ensebmle of systems

    c_ik → p_ik
    assignment probabilities

$$F=\sum_{i,j} c_{ik} E_{ik}=\sum_{i,k} c_{ik} \frac{(z_i-z_k)^2}{\sigma_{z_i}^2}$$

z_i : track z at point of closest approach to beamline
σ_{zi} : z_i uncertainty
z_k : vertex candidate position
c_{ik} : 1, if track is assigned to vertex
0, otherwise



$$p_{ik}=\frac{\rho_k e^{-\frac{(z_i-z_k)^2}{T\sigma_z^2}}}{\rho_0 e^{-\mu^2/T}+\sum_{k'} \rho_{k'} e^{-\frac{(z_i-z_{k'})^2}{T\sigma_z^2}}}$$

1. maximize Entropy for fixed E (←→ T)
   - @ fixed T minimize free energy F wrt cluster positions z_k
   - reaches constellation w/ no net "forces" on clusters (iteratively)
2. when minimum reached, reduce T and repeat until $T=T_{min}$

NB:
algorithm not robust wrt **outliers** that pass track selection
➔ outlier rejection competes w/ splitting, only below T_{min}
➔ allows tracks to be associated to no vertex at all → outlier suppression, like in AVF
➔ clusters with too few tracks are discarded

bibliography

details of the tracking and vertexing performance results