Seed Finding in the ACTS Software Package - Algorithms and Optimizations

Corentin Allaire[1], Françoise Bouvet[1], Noemi Calace[2], **Luis Falda Coelho**[2,3], Hadrien Grasland[1], David Rousseau[1], Andreas Salzburger[2], Stephen Nicholas Swatman[2,4], Carlo Varni[5], Beomki Yeo[5]

[1]IJCLab
[2]CERN
[3]University of Coimbra + LIP
[4]University of Amsterdam
[5]University of California, Berkeley

Connecting The Dots

11 Oct 2023

A Common Tracking Software (ACTS)

Experiment-independent toolkit for charged particle track reconstruction in HEP experiments [arXiv:2106.13593]

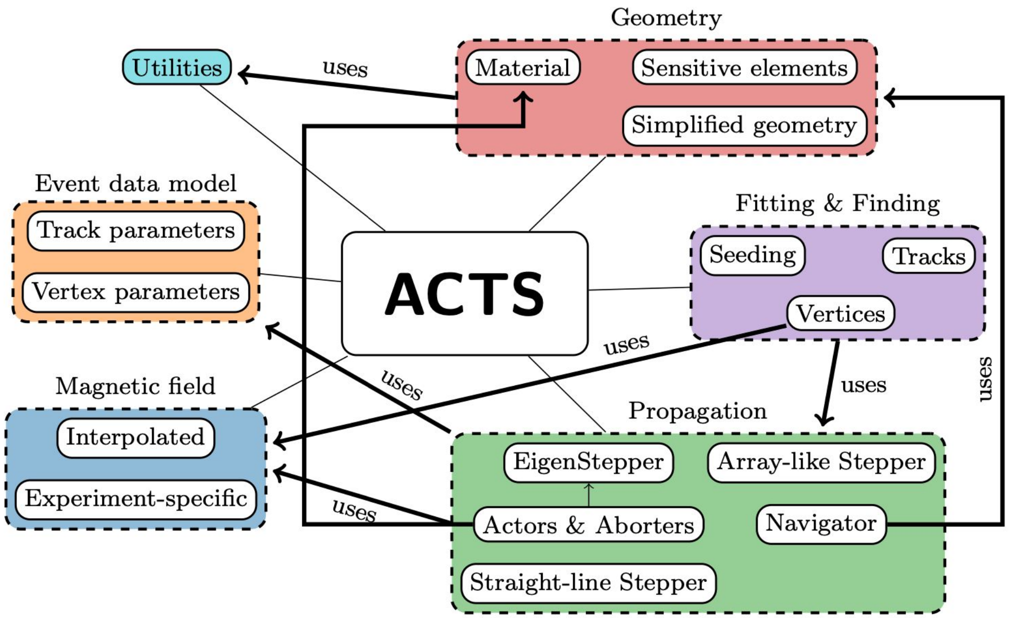
- Designed for modern computing architectures and multi-threaded event processing
- Entirely agnostic to the details of the detection technologies and magnetic field configuration
- Implemented in modern C++ 17 (ACTS GitHub), highly customizable and extendable

CPU-based seeding strategies used by several experiments:

- Default Seeding
- Orthogonal Seeding
- Truth-based (Andi's talk)

R&D platform to explore new techniques:

- GPU parallelisation of seeding
- ML-based seed filtering



CPU Based Seeding - ACTS Default Seeding

CPU Based Seeding - ACTS Default Seeding

The Seeding Algorithm

Essential part of the tracking chain:

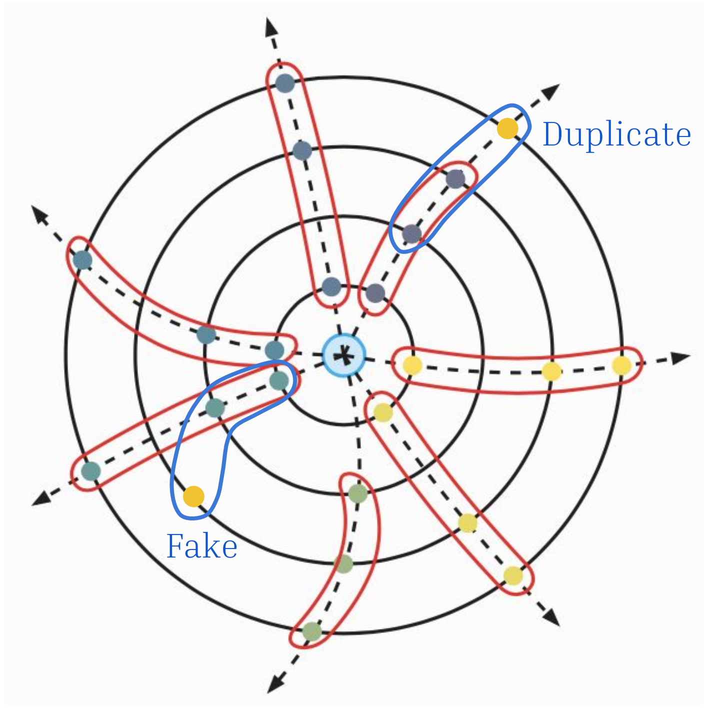
- Track finding uses seeds parameters as first estimate of track direction and momentum

Start by combining 3D representation of detector measurements (SPs)

Finding too many duplicated or fake seeds increase the time needed for tracking

$$\begin{tikzcd}
\boxed{\text{Seed Finder}} \arrow[r] \arrow[d] & \boxed{\text{Seed Filter}} \arrow[r] \arrow[d] & \boxed{\text{Seed Confirmation}} \arrow[d] \\
\text{Build SP combinations} & \text{Filter unwanted combinations} & \text{Keep only the best combinations}
\end{tikzcd}$$

Large number of parameters (some are specific or dependent on geometry) → ACTS can automatically tune the parameter based on tracking performance (Rocky's talk from CTD 2022)



ACTS Seeding Documentation

CPU Based Seeding - ACTS Default Seeding

Seed Finding:

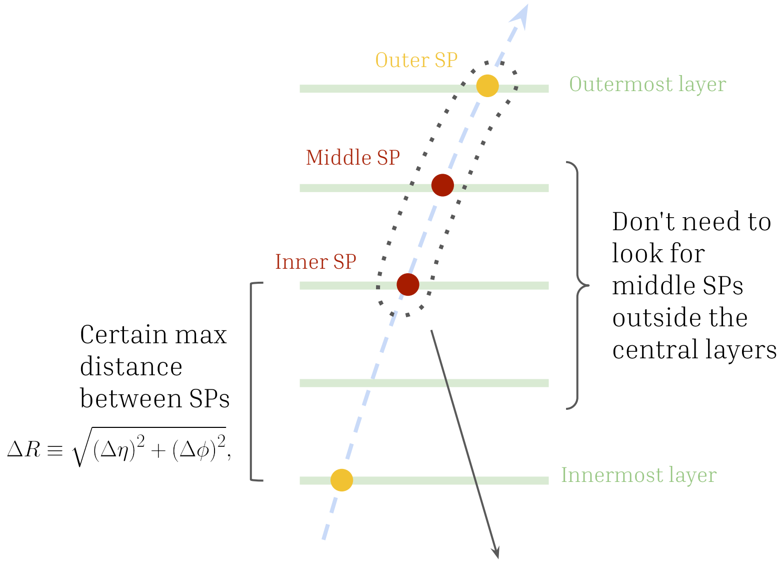
Track seeds are created by combining SPs

- Built assuming helical path from the center of detector in a homogeneous magnetic field
- Starts from selecting a middle SP in a certain detector layer
- Duplets formed from inner and outer SPs:
  - Check if middle SP in region of interest
  - Check compatibility between duplets (ΔR, ΔZ, within collision region)

Seed Filtering:

Various constraint conditions are applied to reduce the number of seed candidates:

- Compatibility cuts between SP-triplets



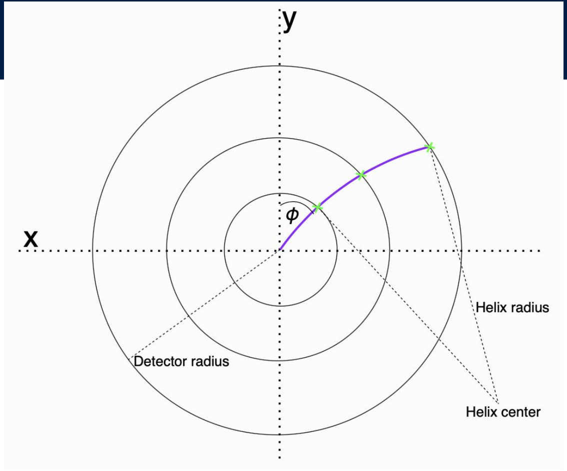
Outer SP
Outermost layer
Middle SP
Inner SP
Don't need to look for middle SPs outside the central layers
Certain max distance between SPs
Innermost layer

$$\Delta R \equiv \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2},$$

Check for curvature compatibility with minimum p_T scattering, transverse impact parameter d_0

$$\boxed{\left(\frac{1}{\tan\theta_b} - \frac{1}{\tan\theta_t}\right)^2 < \sigma^2_{p_T^{min}} + \sigma^2_f,}$$

CPU Based Seeding - ACTS Default Seeding



Seed Confirmation:

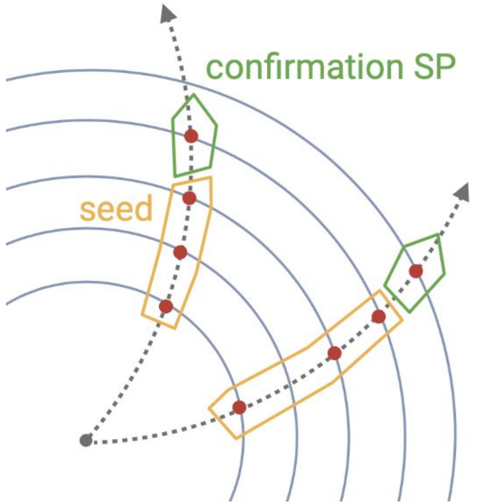
After selecting the SP-triplets, a seed confirmation procedure is applied to all the triplet combinations:

- Compare seeds with similar helix radius
- Rank the seeds based on a customisable weight

$$w = (\underbrace{c_1 \cdot N_t}_{\text{More measurements leads to higher quality}} - \underbrace{c_2 \cdot d_0 - c_3|z_0|}_{\text{Smaller IP} \rightarrow \text{higher probability of track arriving from the interaction point}}) + \textbf{detector specific cuts}$$

Improving the quality of the final track collections by rejecting lower-quality seeds

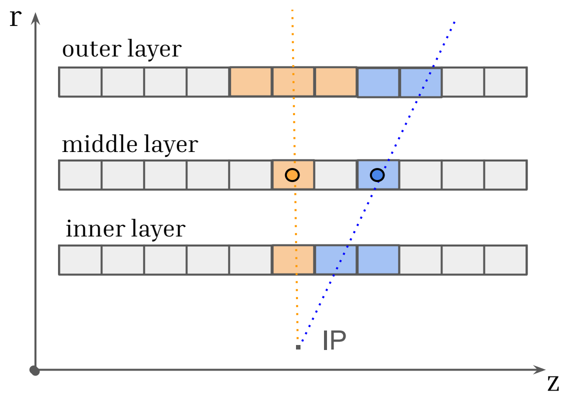
In the end we keep only the best ranked seeds



CPU Based Seeding - ACTS Default Seeding

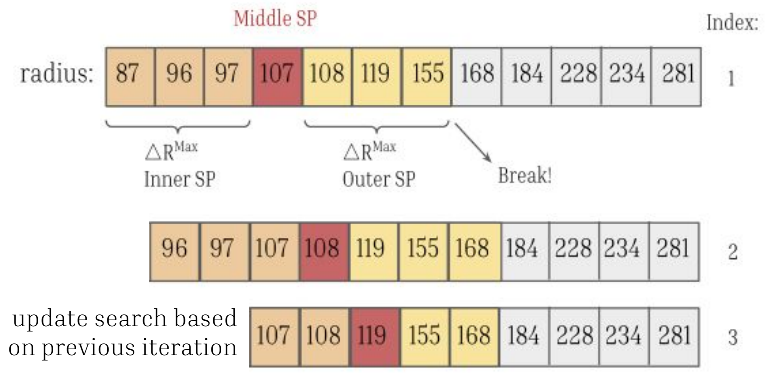
Dealing With High Multiplicity:

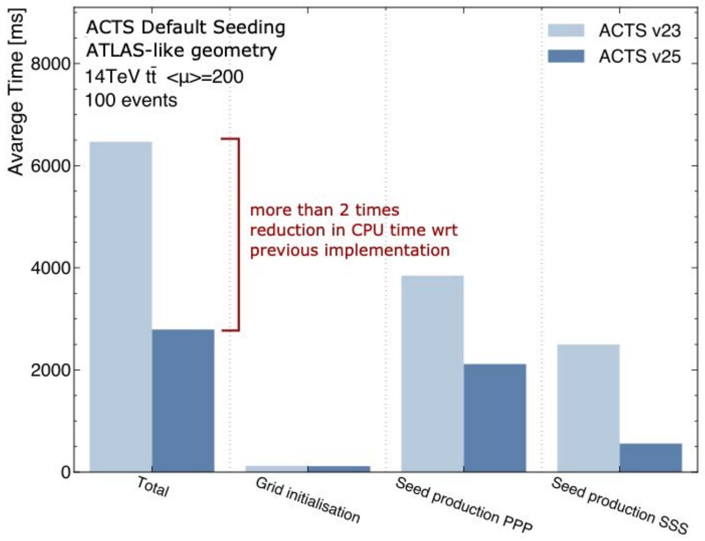
Geometrical assumptions of the curvature of tracks to limit the search to certain neighbouring cells (strongly inspired by ATLAS Phase-II software developments and physics performance studies)



Sort all the SPs in one event based on the SP radius:

- Binary search to avoid looking outside the region of interest when selecting the compatible dublets





Adapting selection cuts, improving the logic, memory allocation, internal classes, hot spots

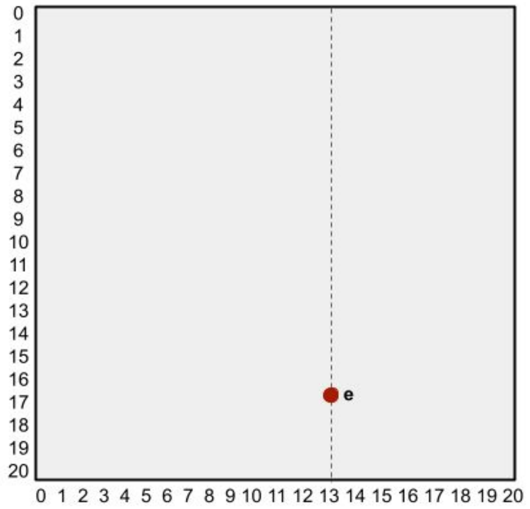
CPU Based Seeding - ACTS Orthogonal Seeding

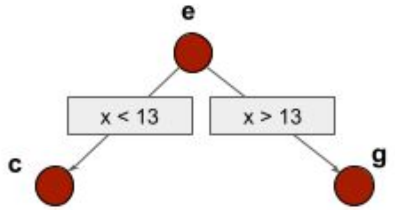
CPU Based Seeding - ACTS Orthogonal Seeding

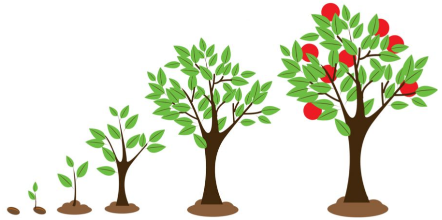
Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces







Splitting is performed along one of the **z-r-φ** dimensions of the SPs, based on median value of that dimension:

- Sorting the range to find the median is too expensive O(n log n). But we can use the middle value between the min and max O(1)
- Dynamic median finding: exact median for small data sets, approximate median for larger data sets
- Balanced KD-tree can improve search performance

This creates two child nodes containing points that are smaller or larger than the median value

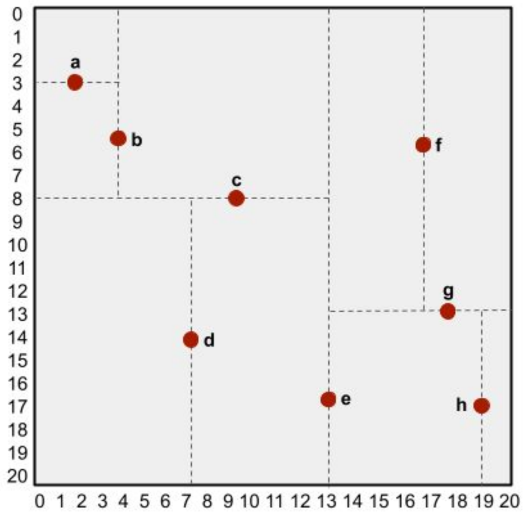
Stephen's talk on ACTS workshop 2022

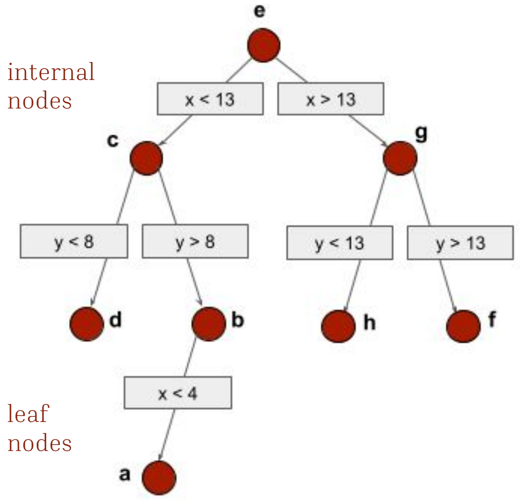
CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces





Splitting is recursively applied to each child node, alternating the dimension along which the split is performed:

- Results in a binary tree structure
- Each node represents a subspace of the data

Range search on the tree for each middle SP:

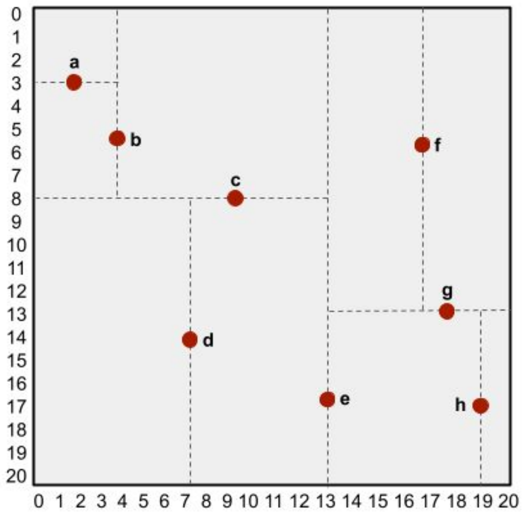
- First we select the search range for that SP based on similar constraints as default seeding
- But now we need to "orthogonalize" this cuts on the tree dimensions

CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces



"Orthogonalizing" means making search range cuts perpendicular to the tree dimensions:

- Trivial for some cases:

$$g_{\min}(\vec{x}) = x_\phi - \Delta\phi_{\max}$$
$$g_{\max}(\vec{x}) = x_\phi + \Delta\phi_{\max}$$

φ should not change too much between SPs

- But not for all of them:

$$z_0^{min} < z_M - r_M \times \frac{\Delta Z}{\Delta R} < z_0^{max}$$

duplet would have originated from the collision point

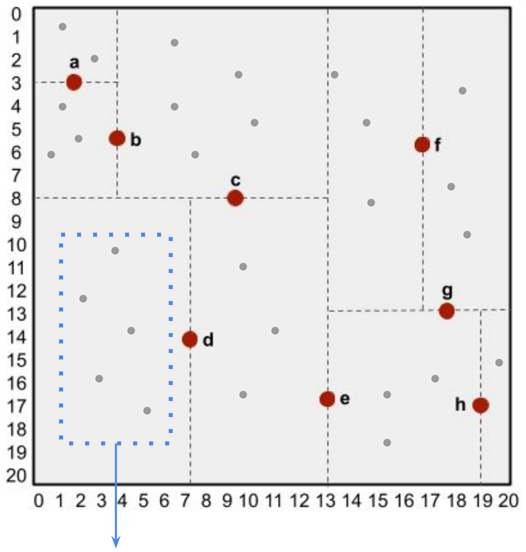
  - Define a weaker version in orthogonal fashion
  - Can always check the tighter constraint in a non-orthogonal way later

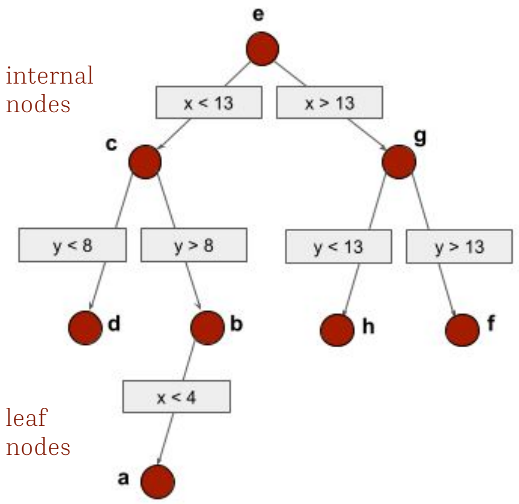
CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces





Bounding Box - It is the smallest box that contains all the points within that node

Performing a query on the tree:

- Algorithm traverses the tree by comparing a certain middle SP to the splitting value of each node
- In case the node's bounding box is fully contained within the region of interest we build compatible SP combinations
- Not necessarily a leaf node - stop the search if reached an internal node that is fully contained by the search range
- If leaf node is not fully contained - we take only the SPs that are

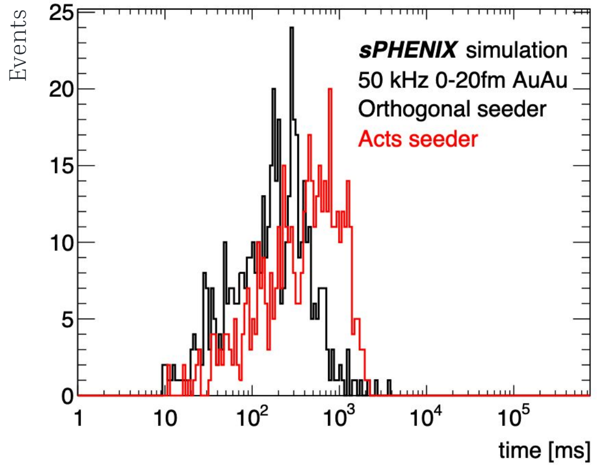
CPU Based Seeding - ACTS Orthogonal Seeding

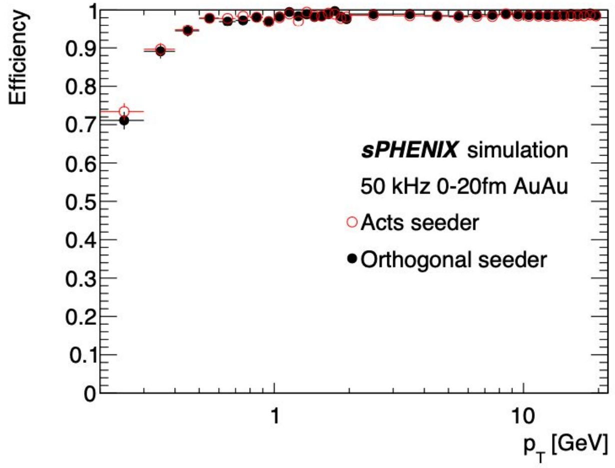
Can be expanded to map points into 6D (**x,y,z,φ,θ,r**) space! More discriminatory power

KD-trees are considered GPU-friendly data structures, which can be efficiently constructed and queried in a massively parallel environment → could be powerful in GPUs!

Good physics performance but still under study:

- May not necessarily be faster than default approach
- Depends a lot on the detector layout and the seeding configuration!





Studies from sPHENIX show similar physics performance and better CPU performance

Also being optimised for ATLAS Phase-II ITk detector

ACTS Seeding Across Experiments

ACTS Seeding Across Experiments

ATLAS:

Ongoing software upgrade program with extensive use of ACTS for Phase-II reconstruction

Implementation and integration of ACTS ITk Tracking chain in ATLAS ([Paul's talk])

At least same physics performance, but better time performance, modernise the software and ensure long-term maintainability

Exactly identical physics performance and similar timing performance with default seeding (testing orthogonal seeding time)

sPHENIX:

Similar occupancies to HL-LHC for Au+Au collisions

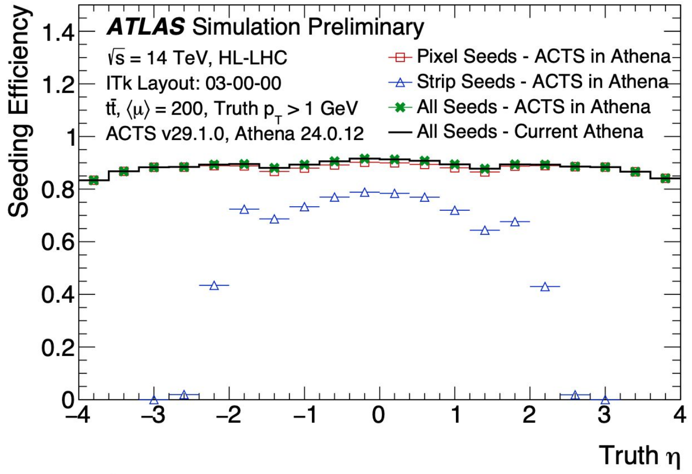
Good performance with ACTS default seeding

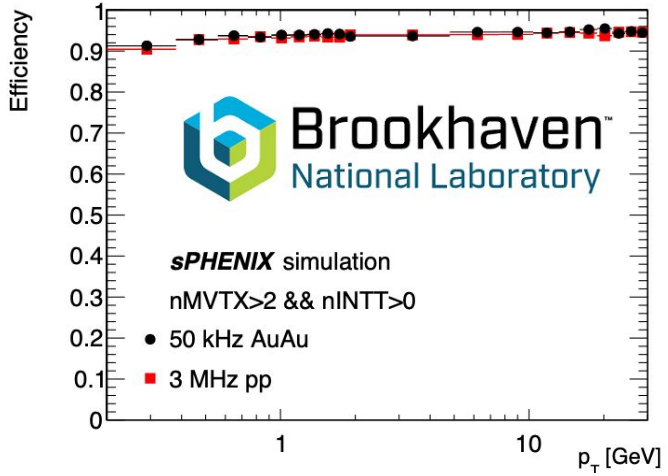
Higher CPU and purity with certain orthogonal seeding configuration but lower displaced track efficiency

ePIC:

Many studies on the effects of tuning the default seed finder and filter on the seeding efficiency and seed multiplicity

ACTS seeding + track finding based on Combinatorial Kalman Filter (CKF) working well for reconstruction of single-particle events in e- p deep inelastic scattering





ACTS Seeding Across Experiments

ALICE:

ALICE upgrade using ACTS default seeding algorithm. Configuration that works well for a Pb-Pb simulation

Optimising processing time, intending to test other approaches and compare the performance

NA60+:

Proposed at CERN SPS to study electromagnetic and hard probes of the quark gluon plasma in heavy ion collisions

High track multiplicity using ACTS. Parameter tuning to run the seeding for a fixed target configuration

Evaluating the expected density of particles and the cuts applied to the seeds

CEPC:

Proposed circular e- p collider for precision measurements of Higgs properties, electroweak physics, flavor physics, exotic decays, BSM …

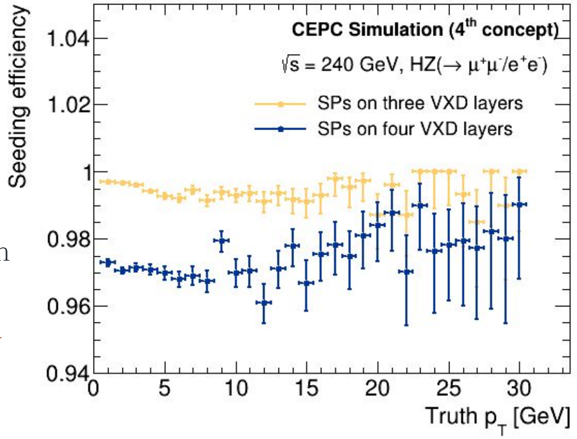
Default ACTS seeding with >99% seeding efficiency if using SPs from >3 VXD layers

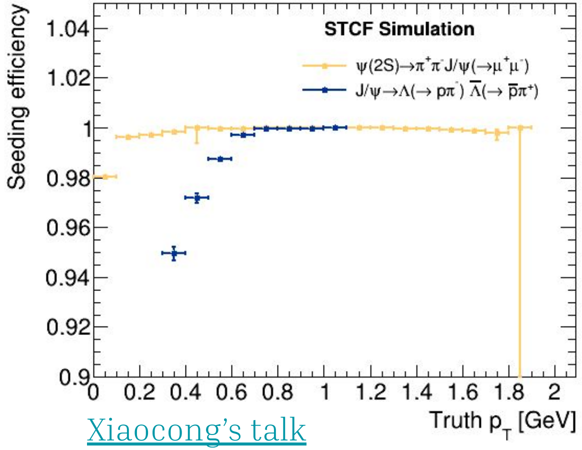
Super Tau Charm Facility (STCF):

Future e+e- collider operating at tau-charm region (2-7 GeV) with peak lumi > 50 times current BEPCII

Default seeding algorithm not optimal for long-lived particle (that leave less hits in the inner tracker)

- Only 3 layers in the inner tracker
- More layers with larger radii might help recover the efficiency loss





Xiaocong's talk

R&D Lines for Seeding - GPU based Seeding

R&D Lines for Seeding with ACTS

Two dedicated R&D lines in ACTS:

- Parallel code execution, mainly focus on GPU accelerators and portability
- Machine learning based/inspired modules

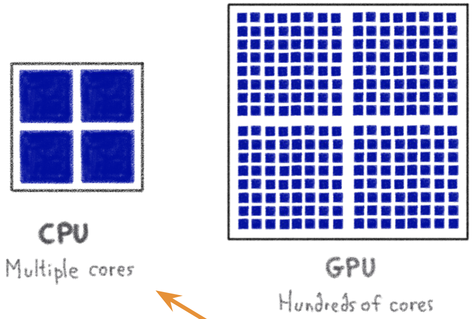
GPU Based Seeding:

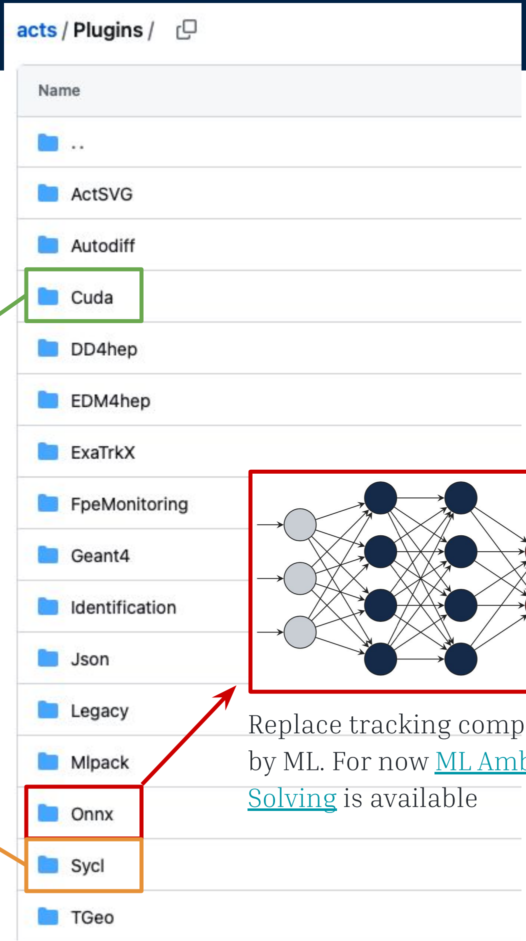
R&D to offload tracking algorithms to GPUs

GPU-based tracking tools with ACTS are available (Seeding, Kalman Filter, geometry navigator)

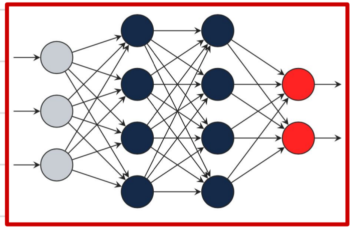
Requirements:

- Same physics performance as the existing CPU algorithms
- Realistic detector setup
- Event Data Model (EDM) shared by CPU and GPU
- Primarily focusing on CUDA and SYCL implementations




acts / Plugins /

Name
..
ActSVG
Autodiff
Cuda
DD4hep
EDM4hep
ExaTrkX
FpeMonitoring
Geant4
Identification
Json
Legacy
Mlpack
Onnx
Sycl
TGeo



Replace tracking components by ML. For now ML Ambiguity Solving is available

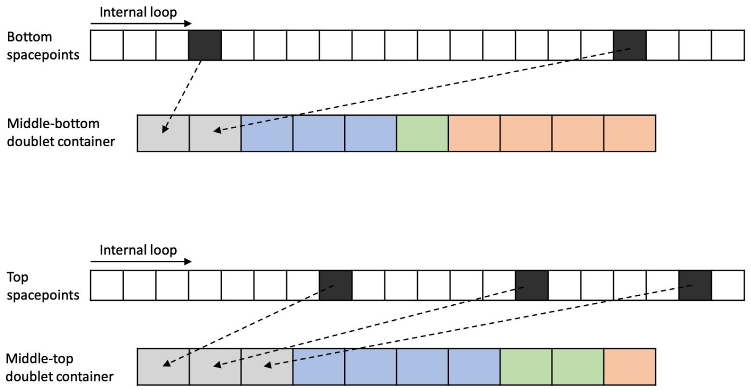
GPU Based Seeding

Similar algorithm as CPU-based seeding:

- But need to count the number of item objects (SPs or doublets or triplets) for each stage of the seeding to properly allocated the workload between the different cores and pre-assign the memory space for the objects

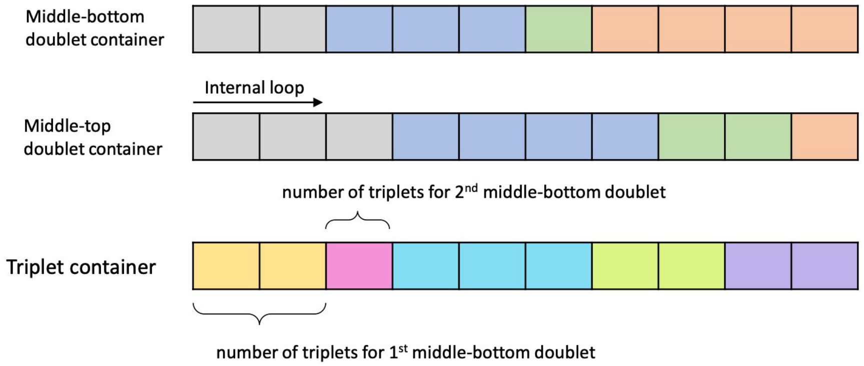
Multiple thread blocks for a certain (φ, z) grid bin of the detector

Dublet Finding:



Every thread (for a compatible middle SP) iterates over inner and outer SPs in neighbour bins to record the doublet objects

Triplet Finding:


Middle-bottom doublet container
Internal loop
Middle-top doublet container
number of triplets for 2nd middle-bottom doublet
Triplet container
number of triplets for 1st middle-bottom doublet

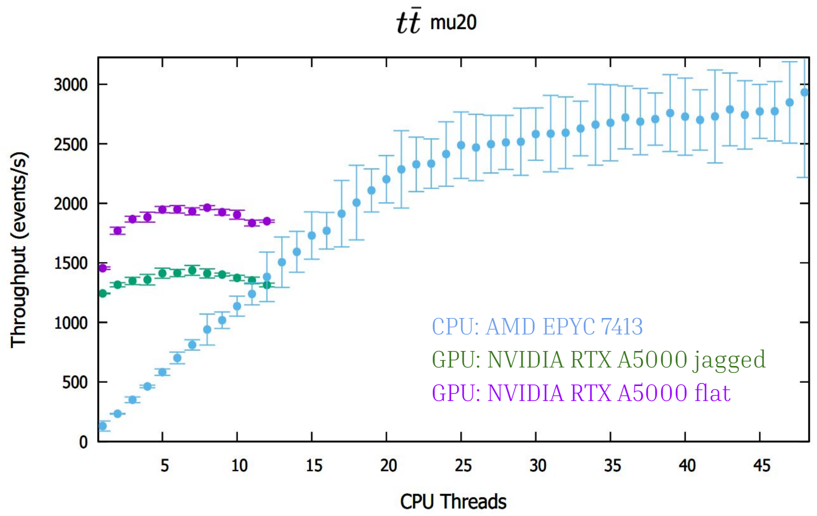
Every thread (for a compatible middle-bottom doublet) iterates over middle-top doublets, whose middle SP is the same, to record the triplet objects

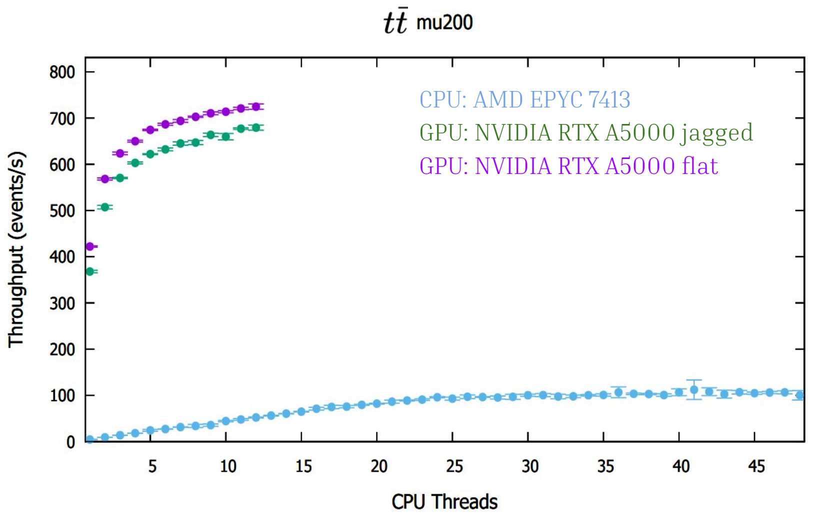
Seed Filter + Confirmation: similar approach

GPU Based Seeding

Seeding with a flat(ter) EDM:

- Seeding requires dealing with nested data structures where inner arrays or vectors can have different lengths for storage of doublets and triplets
- Testing the effect of removing jagged vectors by allocating a single large flat vector
- Seeding with GPU is worthwhile above a certain level of pileup





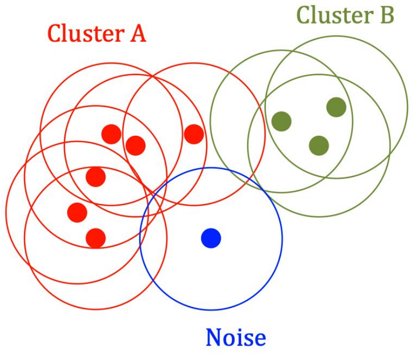
R&D Lines for Seeding - ML Seed Filter

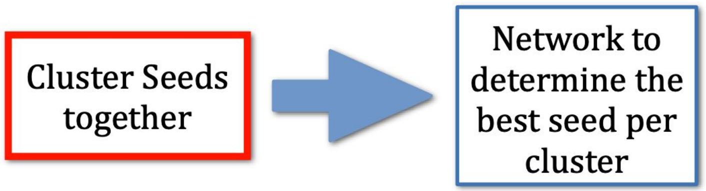
Machine Learning Based Seed Filtering in ACTS

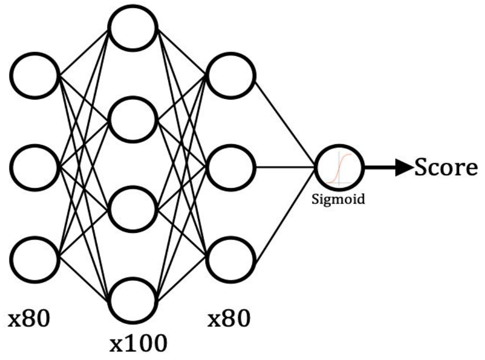
During track reconstruction, many more tracks than truth particles can be reconstructed due to high number of seeds

Reducing the duplicate rate could directly improve the speed of the tracking chain

Can we filter seeds based on a ML algorithm before reconstructing the tracks?







DBScan clustering algorithm used to cluster seeds together in a 4D space using (Phi, Eta, Z_0, p_T), ideally 1 cluster ~ 1 truth particle

NN trained is then used to score each seed, keep the highest score per cluster

Input parameters :

- Pt
- Eta
- Phi
- Z0
- Seed Quality
- Space point 1 (x,y,z)
- Space point 2 (x,y,z)
- Space point 3 (x,y,z)

longitudinal impact parameter

Same approach as Corentin's ML based Ambiguity Solving at CTD 2022

Machine Learning Based Seed Filtering in ACTS

Ranking Neural Network:

Training per truth particle basis:

- For each particle look at all associated seeds
- Loss obtained by comparing score of closest seed to truth and score of fakes and duplicates
- Margin Ranking Loss:

$$loss_{part} = \frac{1}{N_{seeds}} \sum^{seeds} max(0, \underset{\text{seed score}}{x} - \underset{\text{good seed score}}{y} + margin)$$

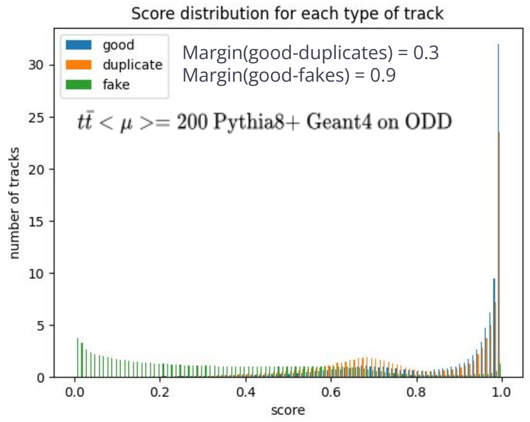
$$\text{score}_{good} - \text{score}_{bad} > \text{margin} \rightarrow loss_{part} = 0$$

Model has correctly ranked the good seed!

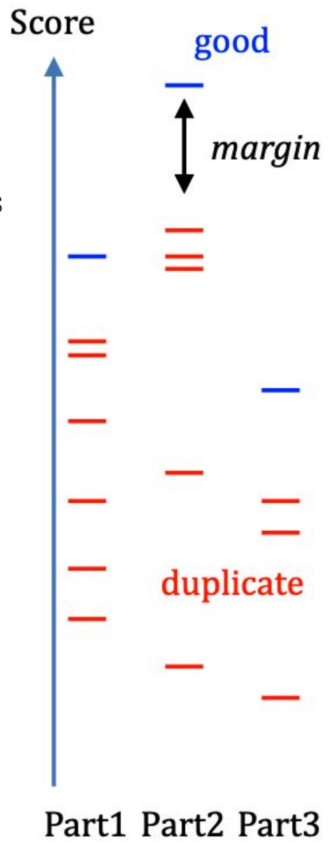
$$\text{score}_{good} - \text{score}_{bad} < \text{margin} \rightarrow loss_{part} > 0$$

Encourage the model to update its parameters

Ensuring that good seeds are ranked higher than fakes and duplicates by at least a certain margin



NN is able to distinguish between fake/duplicate/good seed



Machine Learning Based Seed Filtering in ACTS

ML Seed Filtering:

No parameter tuning needed, only a short training

Available in ACTS via Onnx

Track finding speed-up (x1.5 faster)

Reduction of the number of tracks per event (÷2.6) at the CKF level

Basically same efficiency

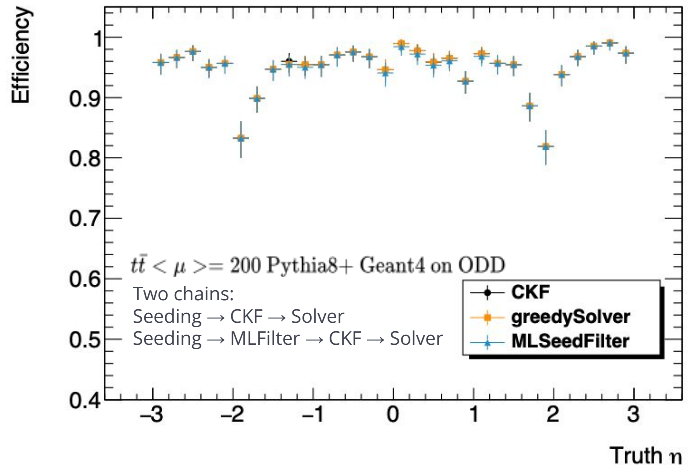
Next Steps:

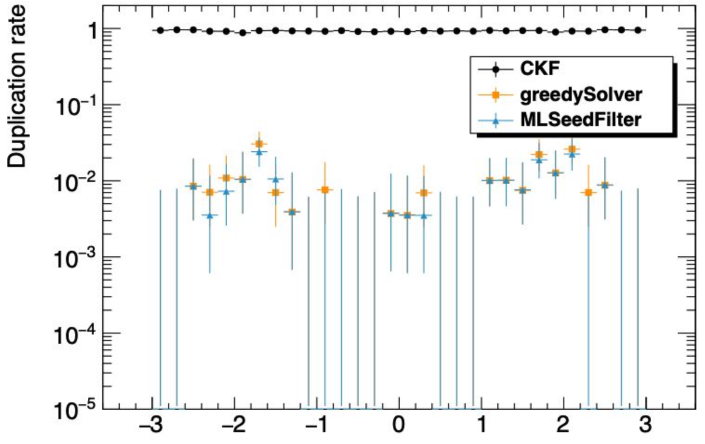
Test and optimise the performance on a more realistic detector

Looking at using more track parameters that we give as input to the CKF based on the reconstructed seeds

We consider the good seed to be the one closest to the truth for training. We would like to use the one that leads to the best track instead:

- Re-feeding back the track reconstruction output as feedback to the NN to improve performance





Summary

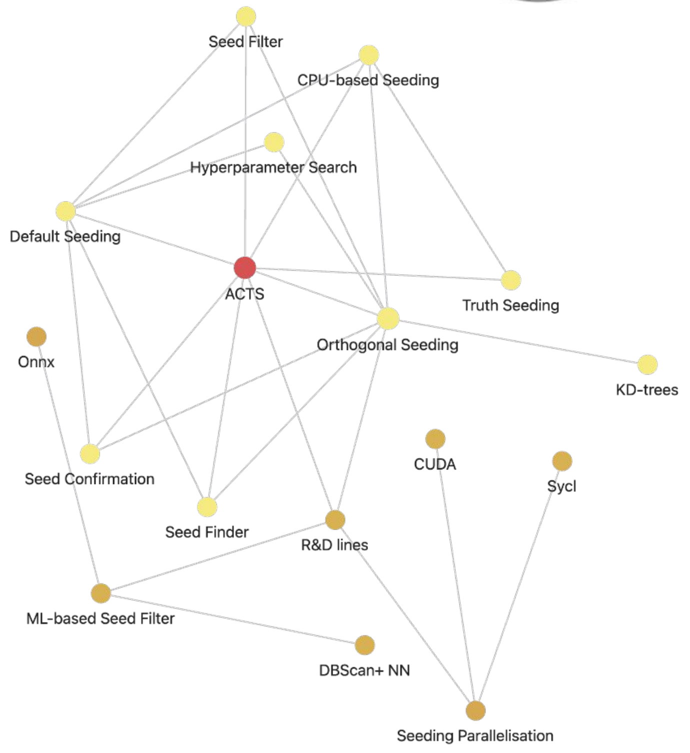
ACTS incorporates CPU-based seeding algorithms:

- A lot of work is necessary to customise the seeding to achieve optimal performance
- ACTS seeding has shown to be quite customisable and perform well:
  - Exceptional physics and time performance across numerous experiments
- Consistent effort towards enhancing flexibility and performance

Two R&D paths for seeding:

- Optimizing code for parallel processing, with a primary emphasis on GPU acceleration and portability
- Application of machine learning techniques in Seeding

Both yielding promising results



Seed Finding in the ACTS Software Package - Algorithms and Optimizations

Corentin Allaire[1], Françoise Bouvet[1], Noemi Calace[2], **Luis Falda Coelho**[2,3], Hadrien Grasland[1], David Rousseau[1], Andreas Salzburger[2], Stephen Nicholas Swatman[2,4], Carlo Varni[5], Beomki Yeo[5]

[1]IJCLab
[2]CERN
[3]University of Coimbra + LIP
[4]University of Amsterdam
[5]University of California, Berkeley

Connecting The Dots

11 Oct 2023

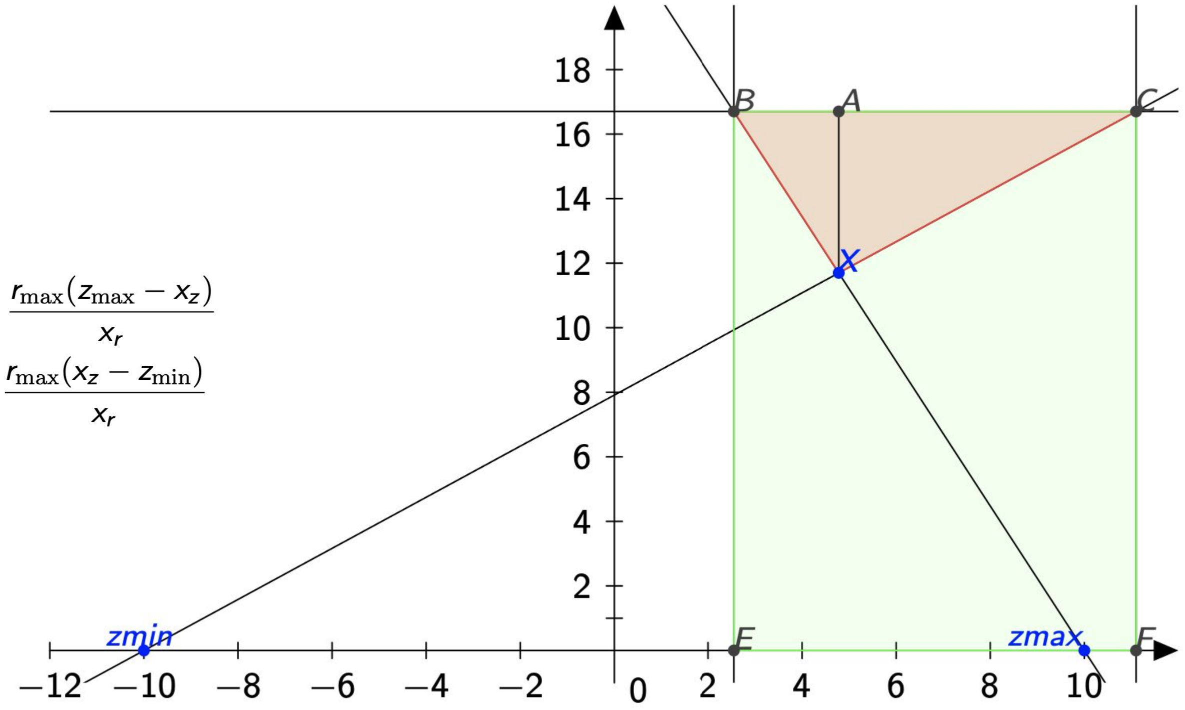
Backup

Approximating the z-intercept

Weakened version can be found by reasoning backwards: instead of starting with two points and checking whether they intersect within a z boundary, start with the z boundary and one point, check where the second point may be

$$g_{\min}(\vec{x}) = z_{\max} - \frac{r_{\max}(z_{\max} - x_z)}{x_r}$$

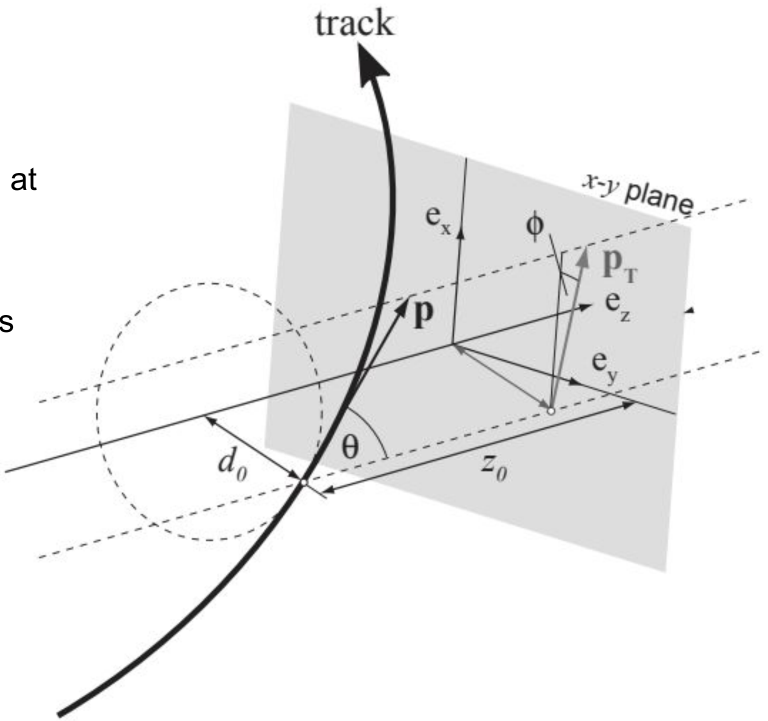
$$g_{\max}(\vec{x}) = z_{\min} + \frac{r_{\max}(x_z - z_{\min})}{x_r}$$



Perigee Representation

- Tracks are described by five parameters and a reference point
- Transverse and longitudinal impact parameters (d_0 and z_0)
- Azimuthal angle φ and the polar angle θ of the track momentum at the reference point
- Charge of the reconstructed track divided by the magnitude of its momentum
- The reference point used is the average position of the pp interactions

$$\left(d_0, z_0, \phi, \theta, \frac{q}{p}\right)$$



Seeding Cuts

- Cuts are applied on every seed candidate and its components
- Cut on the compatibility between the slope of two seed segments
- Comparing the squared difference between slopes, and comparing to the squared uncertainty in this difference

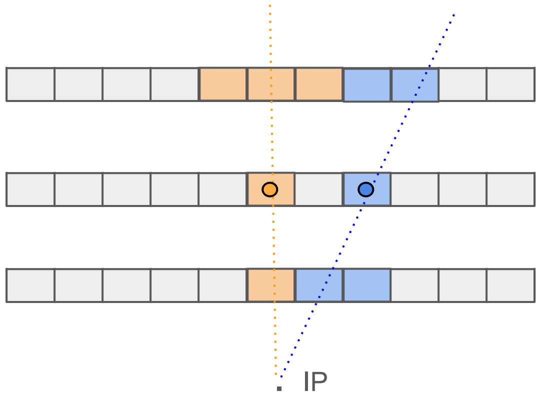
$$\left(\frac{1}{\tan\theta_b} - \frac{1}{\tan\theta_t}\right)^2 - \delta^2 > \sigma^2_{p_T^{min}}$$

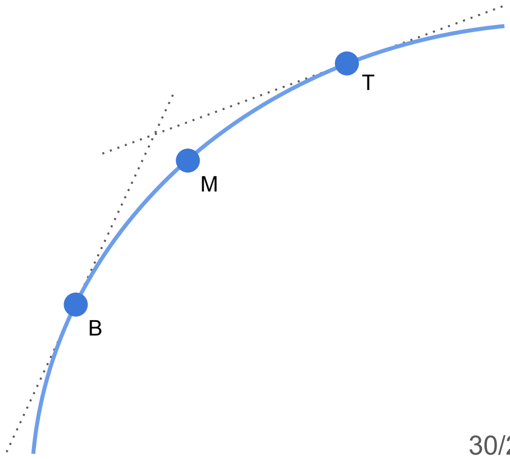
- Include a multiple scattering term assuming the lowest pT allowed

$$\sigma^2_{p_T^{min}} = \left[\left(\frac{1}{\tan\theta_b}\right)^2 + 1\right] \frac{\sigma^2_{scattering} \cdot (Highland)^2}{(p_T^{min})^2}$$

$$Highland = 13.6 \cdot \sqrt{x}\,(1 + 0.038\log(x))$$

Coulomb Scattering





Seeding Cuts

- Exploit the conformal transformation to apply more cuts
- Three SPs belonging to a circle with radius R and center at (x0, y0)
- Assuming that the circle goes through the origin:

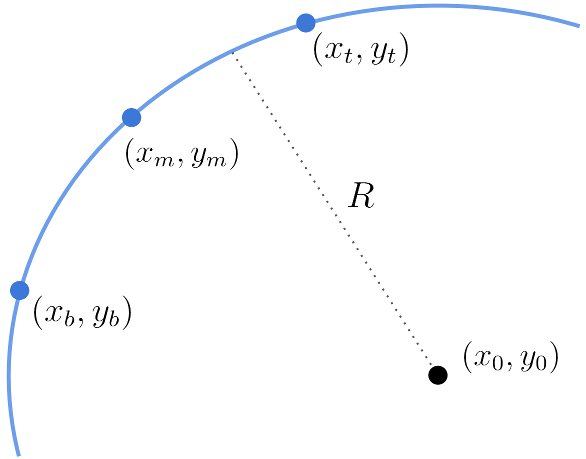
$$v = -\frac{x_0}{y_0}u + \frac{1}{2y_0} = Au + B$$

- Experimentally obtain A and B:

$$A = \frac{v_t - v_b}{u_t - u_b}, \qquad v_b = Au_b + B \Rightarrow B = v_b - Au_b.$$

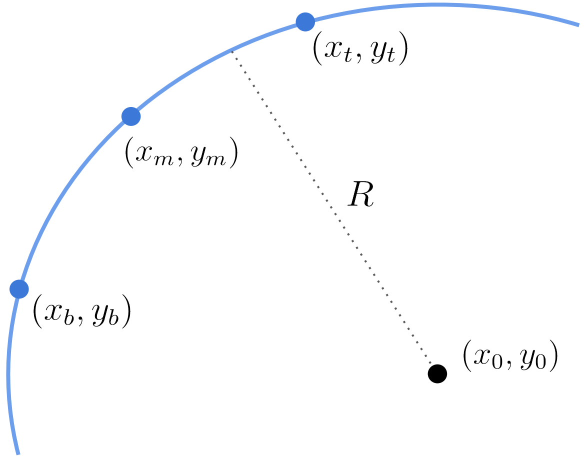
- Apply the pt cut directly on the estimate of R without extra conversions

$$x_0 = -\frac{A}{2B} \quad y_0 = \frac{1}{2B}. \qquad x_0^2 + y_0^2 = R^2 \longrightarrow \boxed{R^2 = \frac{(A^2+1)}{B^2}}$$



Seeding Cuts

- Cut on the estimated impact parameter of the seed
- Refinement of the first tangent compatibility cut, replacing the scattering contribution which assumes the lowest pT by one based on the actual estimated pT
- Cuts on the individual SPs (such as z, R, ΔR, η, compatibility with IP)



Machine Learning Based Seed Filtering in ACTS

DBScan (a classic clustering algorithm) mlpack C++ implementation of many ML algorithms, really old (16+ years) still maintained to this day!

Clustering can be fine tuned to affect efficiency/purity balance

Use 2 parameters :

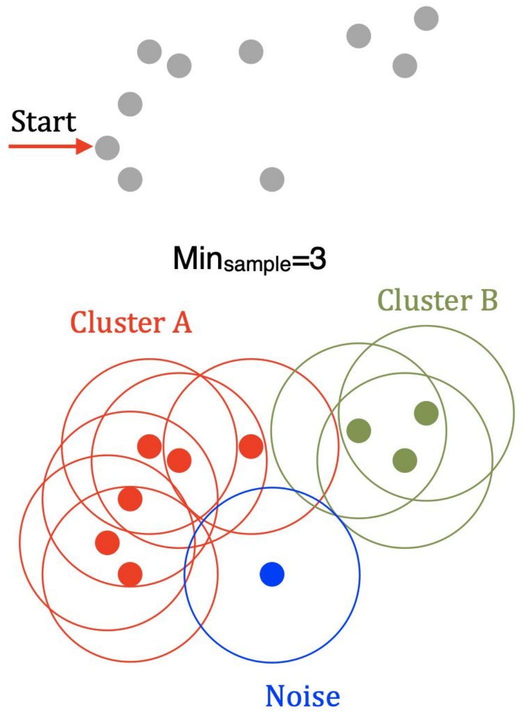
- ε: Max distance between neighbour
- $\mathrm{Min}_{\mathrm{sample}}$: Min number of elements per cluster

More than $\mathrm{Min}_{\mathrm{sample}}$ neighbour ➡ Create a cluster

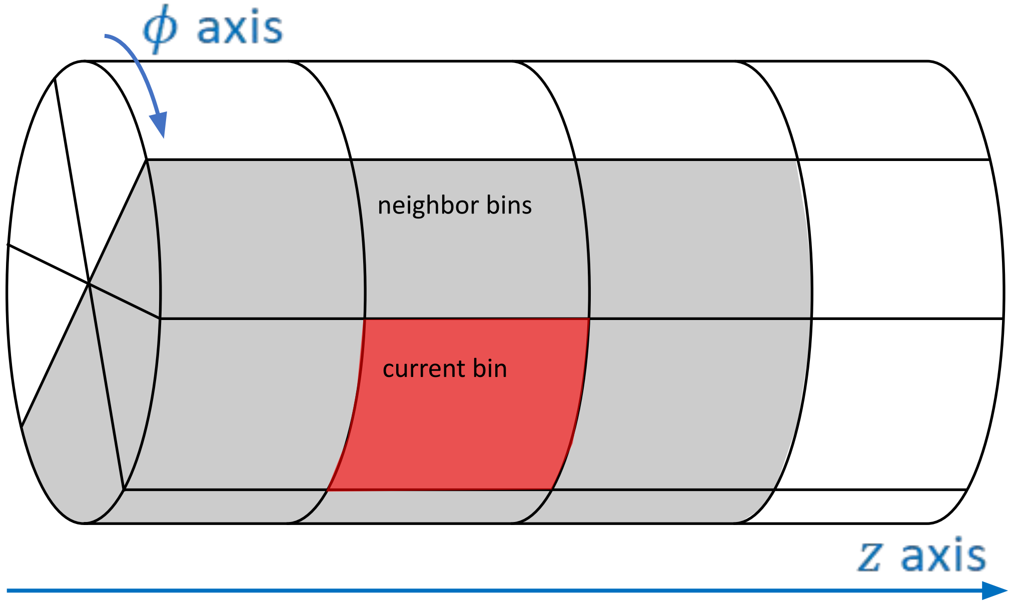
For each element of the cluster, do the same ➡ extend the cluster

In the Seed Filter :

- distance in (η,ϕ); ε=0.07 ; $\mathrm{Min}_{\mathrm{sample}}$=2



Spacepoint binning


ϕ axis
neighbor bins
current bin
z axis