Auto-tuning capabilities of the ACTS track reconstruction suite

Corentin Allaire[1], Rocky Bala Garg[2], Hadrien Benjamin Grasland[1], Elyssa Frances Hofgard[2], Andreas Salzburger[3], Lauren Alexandra Tompkins[2]

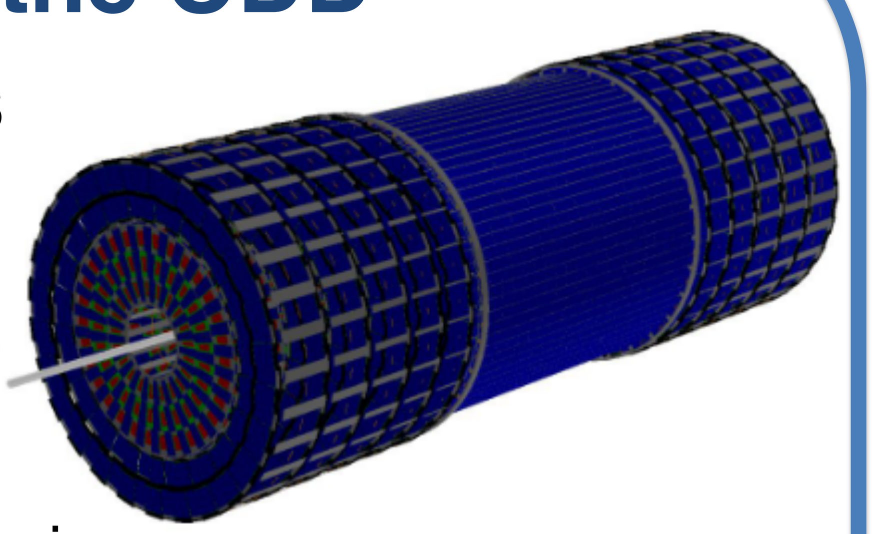
[1]Université Paris-Saclay, [2]Stanford University, [3]CERN

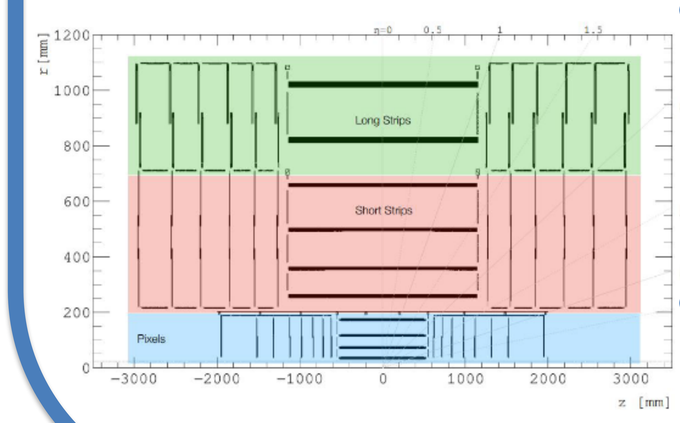
Context and Motivations

- Most tracking algorithm use multiple parameters to account for experimental conditions :
  - Detector geometry
  - Material configuration
  - Pile-up
  - Collision type
  - Center-of-mass energy
  - Many other factors
- Hand-tuned parameters :
  - Trial and tested method
  - Provides good configurations
  - Slow (require expert)
  - Long term maintainability issue
  - Expensive retuning
- Auto-tuned parameters :
  - Faster : trade CPU time for human time
  - Easier to rerun : "change the conditions and press the button"
  - Can be performed in parallel
  - More granular : Sub-detector level optimisation

ACTS and the ODD

- Studies performed within ACTS (A Common Tracking Software) :
  - Open source tracking software
  - Experiment independent (ATLAS, FASER, sPHENIX, EIC,…)
- Implement most tracking algorithms and a full tracking chain





- Used the ODD (Open Data Detector) :
  - Virtual detector
  - Full silicon design (similar to the ATLAS ITk)
  - Used in Track ML challenge
- Great environment to develop and test new machine learning based algorithms

Optimisation framework

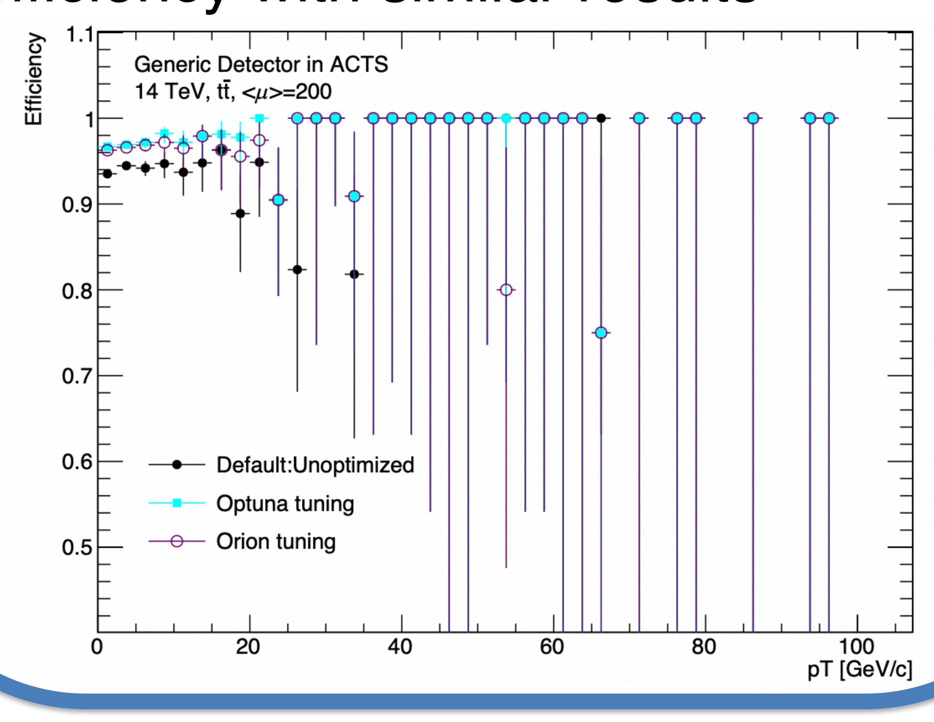
- Optimiser :
  - Test different configuration parameters values for the algorithm
  - For each compute a performance based score
  - Try to find the set of parameters minimising the score
- Tried two different frameworks :
  - ORÍON : asynchronous framework for black-box function optimisation
  - OPTUNA : Open source software for automatic hyper-parameter search
  - Many different optimisation algorithm for each framework : Random search, ASHA, Tree-structured Parzen Estimator, …
  - Derivative-free approach ➔ works well with high evaluation cost and irregular score function

Seeding

- Create groups of measurement used to seed the track finding
- Optimiser looks for the parameters that maximise this score :

$$Score = Efficiency - \left(FakeRate + \frac{DuplicateRate}{K} + \frac{RunTime}{R}\right)$$

- Efficiency : particle without seed ➔ particle lost
- Fake rate : fake tracks ➔ worse physic performance
- Duplicate rate : Many seeds per track ➔ slower reconstruction
- Both optimisers (~ 1 h) improve efficiency with similar results

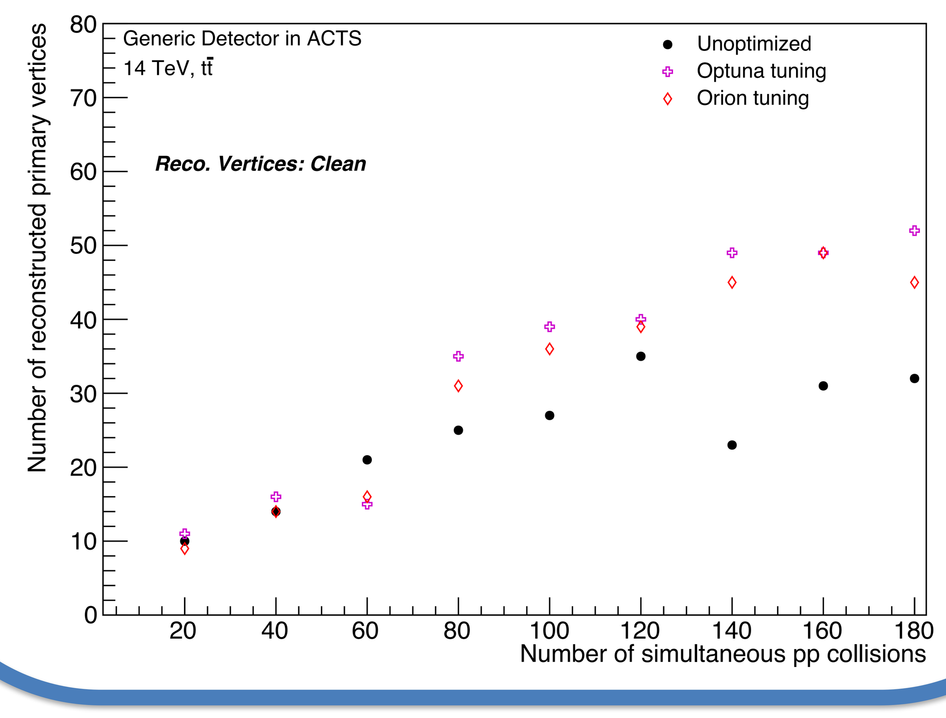


Vertexing

- Combine the informations on all the tracks to find back the interaction points (vertices)
- Use the optimiser to find the parameters that maximise this score :

$$Score = (Eff_{Total} + 2Eff_{Clean}) - (Merged + Split + Fake + Resolution)$$

- Clean : vertex associated with 1 truth particle
- Merged : 1 vertex multiples particles
- Split : multiple vertices one particle
- Reconstruct cleanly as many vertices as possible
- After the optimiser (~ 4 h) : more clean vertices and less fake ones

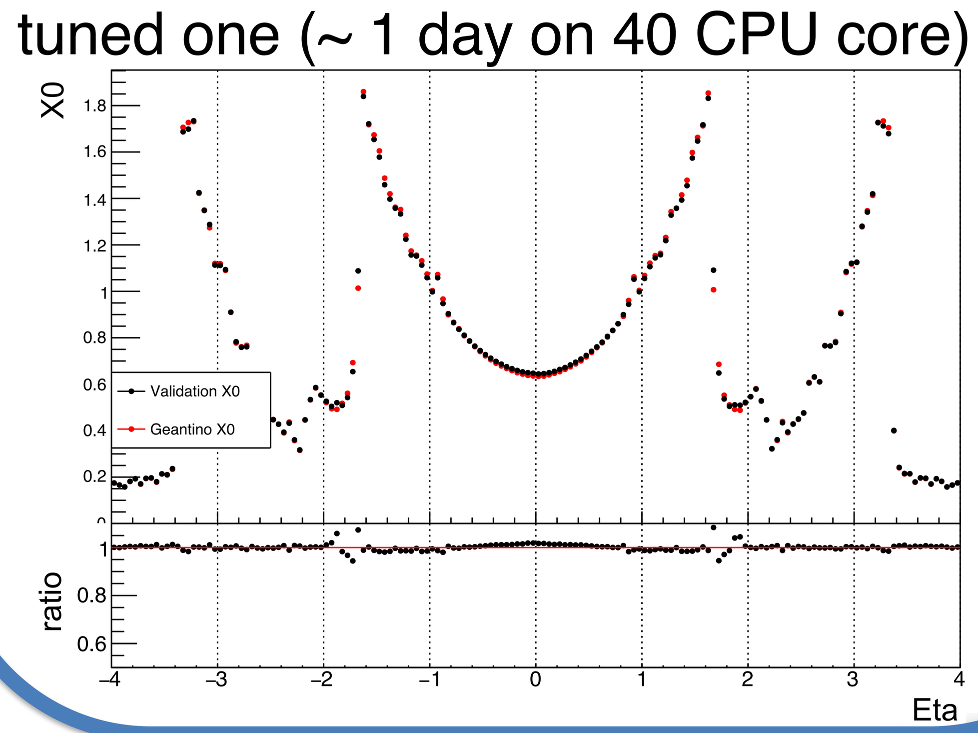


Material Mapping

- Create a simplified material representation for the navigation (100 acts surfaces vs)
- Project detector material onto binned surfaces
- Minimise material variance in each bin :

$$Score = \frac{1}{bins} \times \sum_{bin} variance_{mat} \times (1 + \sqrt{bins})$$

- Map compared by navigating through the detector and collecting the encountered material
- Auto-tuned map as good as the hand-tuned one (~ 1 day on 40 CPU core)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004761.

Poster Session ACAT 2022 - October, 2022