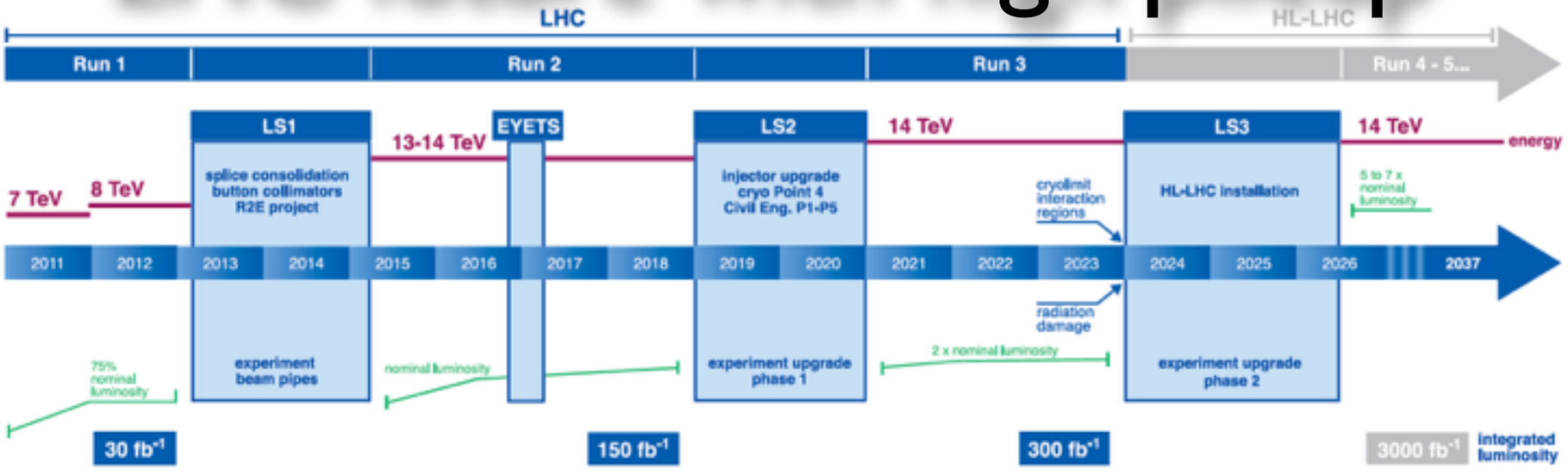
Impact of tracker layout and algorithmic choices on cost of computing at high pileup

G. Cerati, V. Krutelyov, M. Tadel, F. Wuerthwein, A. Yagil

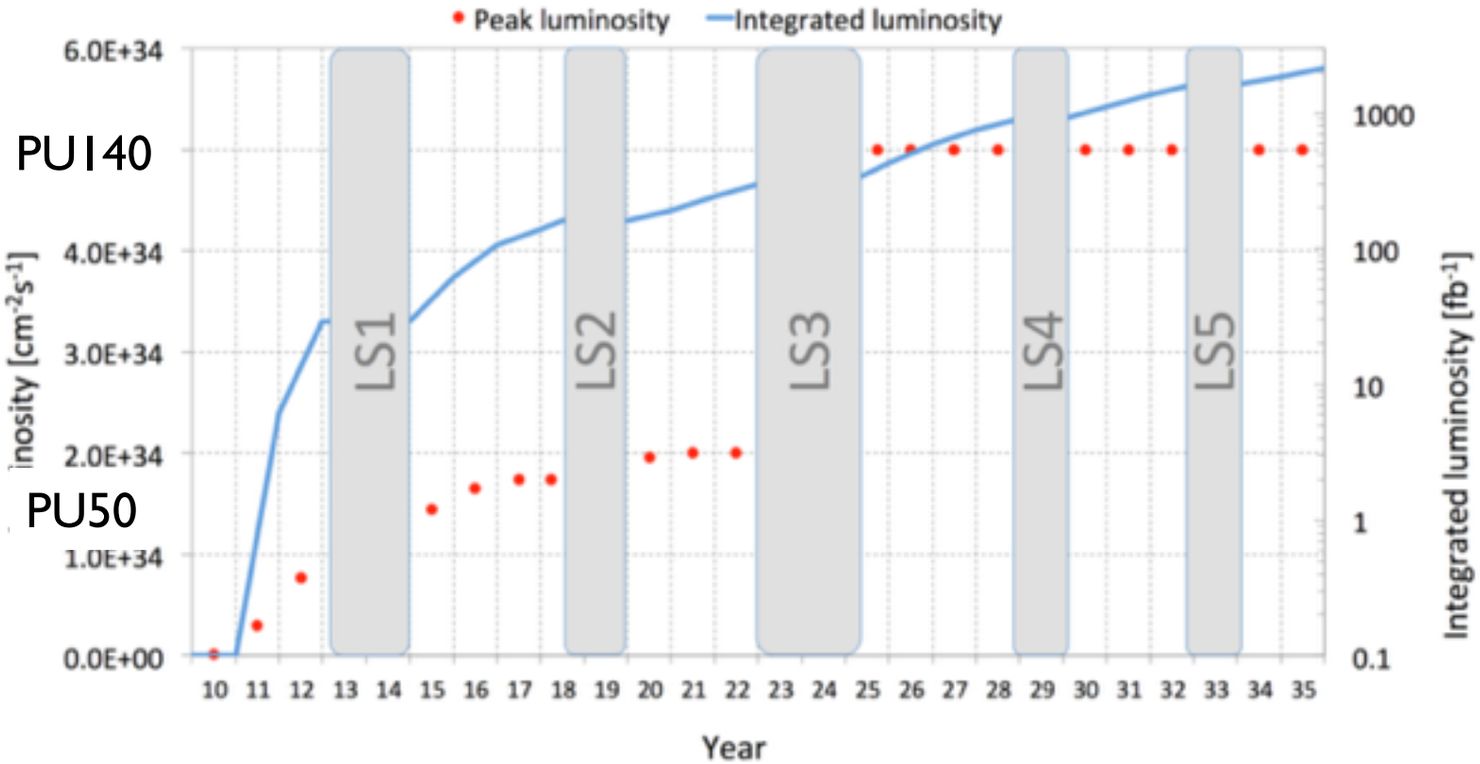
ICHEP 2016
August 6, 2016

- LHC future with high pileup
- Tracking: comparable costs of detector and computing
- Conventional tracking: Kalman filter (KF) based approach
- Evolution to adapt to high pileup
  - Improved use of computing hardware
  - Adjusting thresholds $\Leftarrow$ price in physics
  - New faster algorithms
  - Tracker design to aid track reconstruction
- Example of possible synergy: layer layout impact on reconstruction

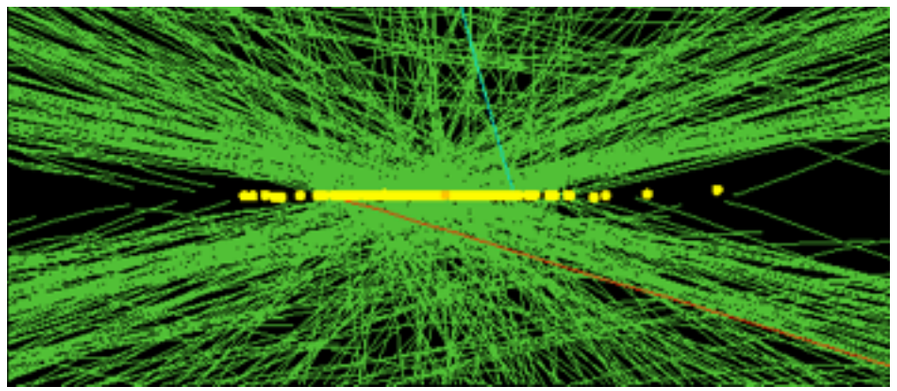
LHC future with high pileup



- Over the next two decades we will see substantial increase in LHC luminosity, which comes with increased event complexity from increasing pileup
- In HL-LHC operations pileup (PU) could reach mean of 200
- Experiments are preparing to upgrade their detectors, most notably, trackers

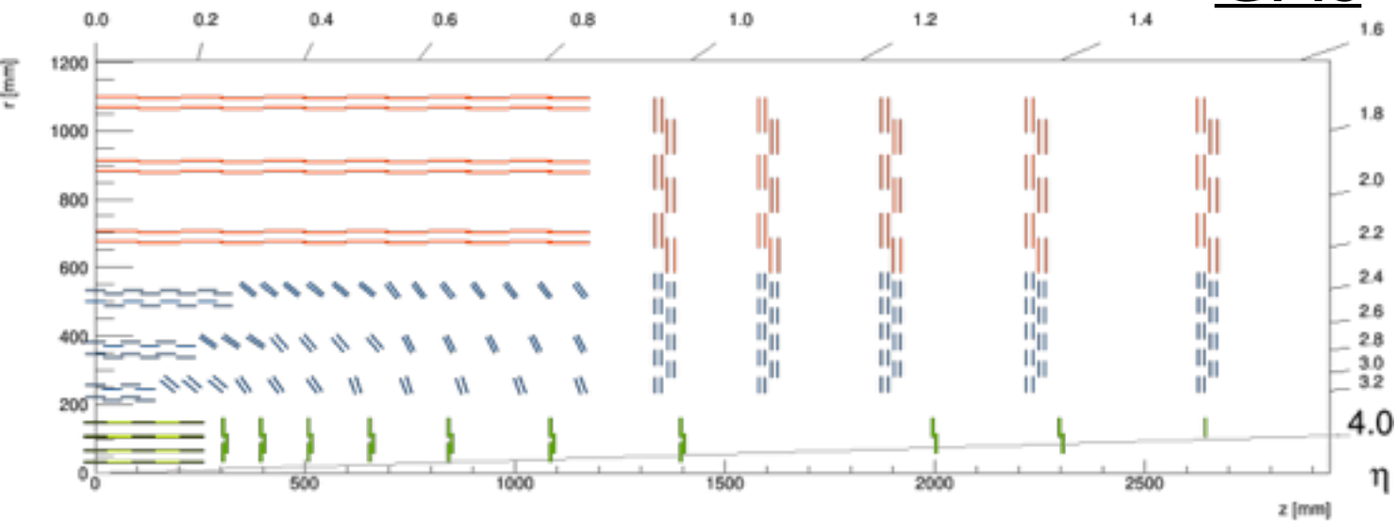


PU~100 actual event with 78 vertices
CMS-PHO-EVENTS-2012-006

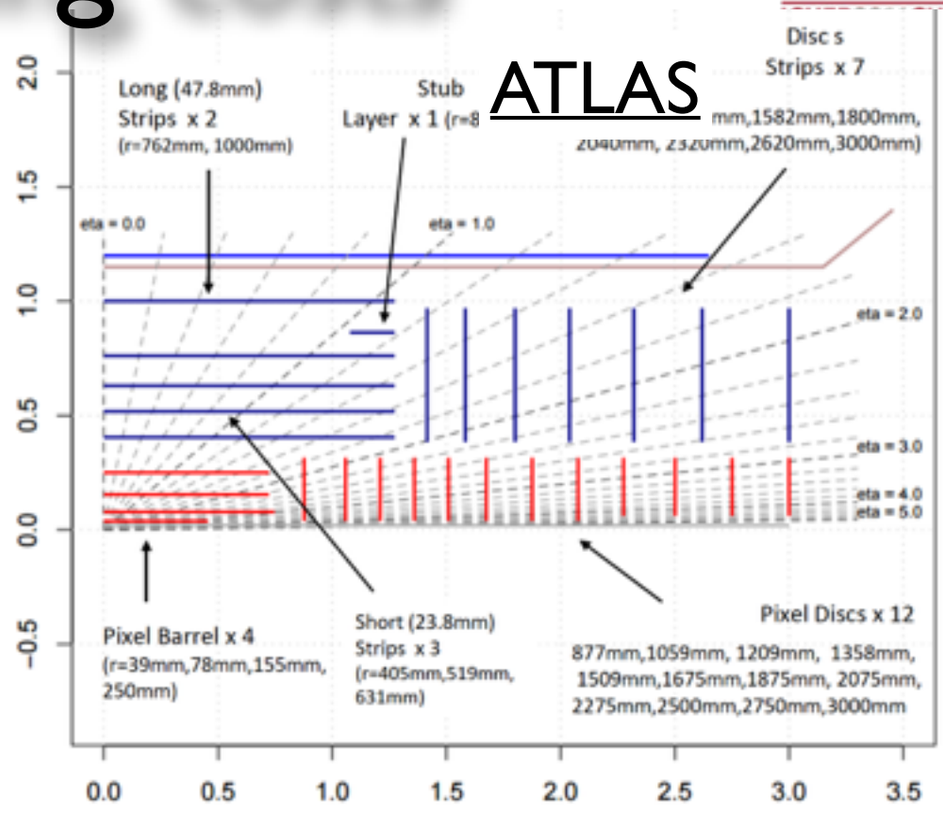


Tracker and Tracking costs

CMS

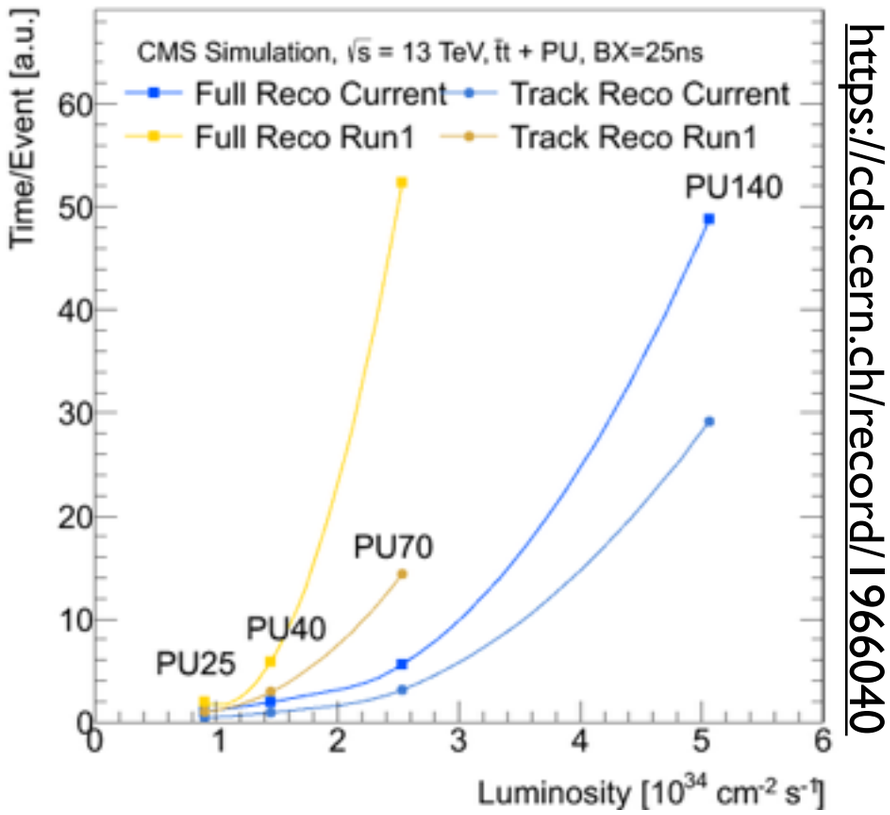


ATLAS



- Somewhat similar tracker designs

- Processing time grows exponentially with increasing pileup
  - *A large fraction of this is in tracking*
- Detector cost is similar to the cost of running track reconstruction over the time of HL-LHC
- Future tracker design decisions should consider both the detector hardware cost and the track reconstruction costs

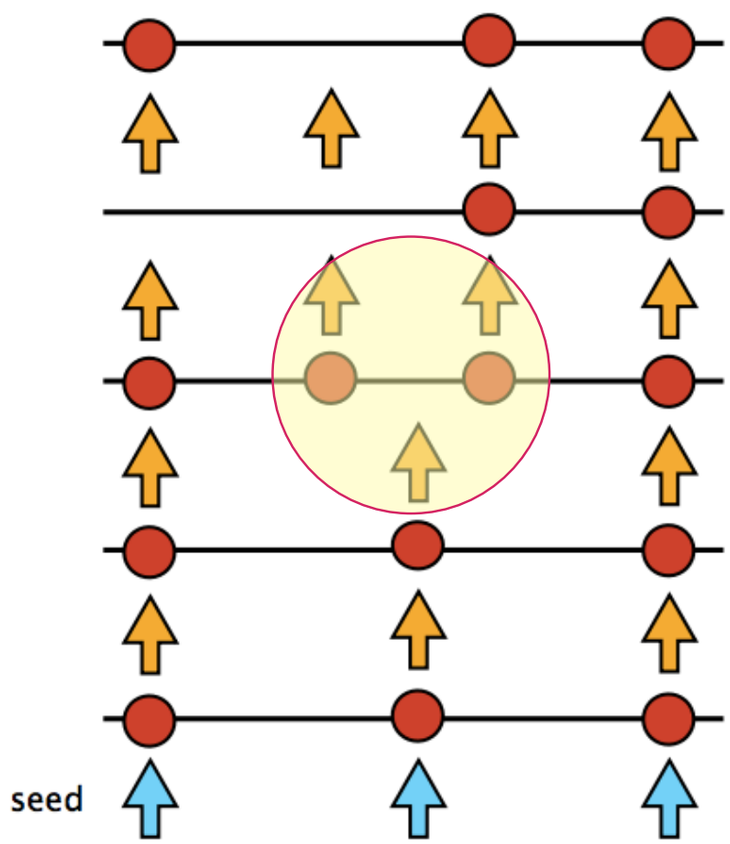


https://cds.cern.ch/record/1966040

Conventional Tracking: KF

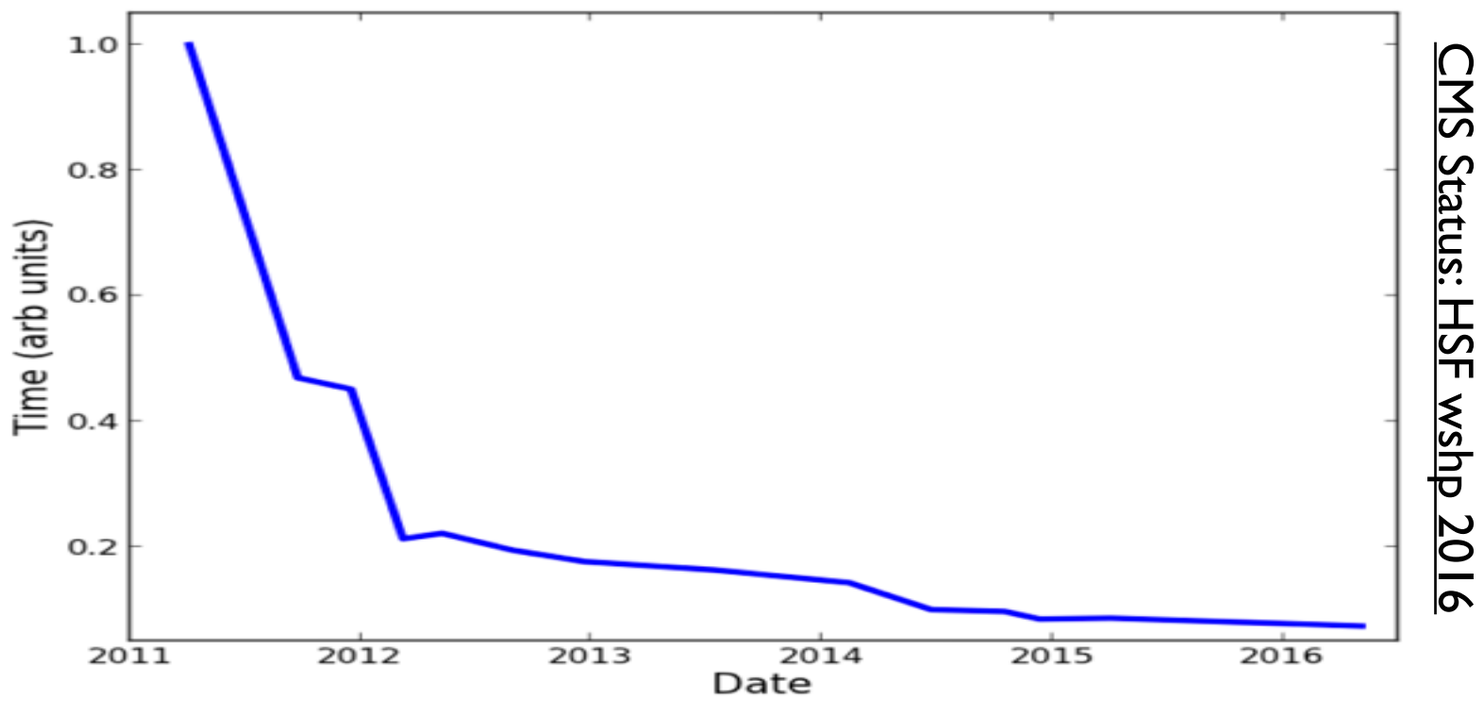
- Tracking is naturally divided in two parts pattern recognition and final track fit
- Kalman Filter (KF) based pattern recognition gives mathematically most optimal pattern selection
- Full track parameters in successive measurement layers are used to make progress. **This is computationally intensive.**
  - **5 parameters and a 5x5 covariance matrix with transformation Jacobians and matrix multiplications and inversions are typical objects and operations needed at each step**
- In high pileup environment track pattern recognition is combinatorially intensive. Combined with computational complexity of KF, tracking takes a lot of time
- Final track fit or final cleanup steps are not combinatorially intensive, they are fast and are justifiably done with KF

New candidate for each ambiguity
Candidates carried to the end
Decide at the end which is best



Evolution to adapt to high pileup

- Ways to address computing resource constraints with increasing pileup (PU)
  - **Improved use of computing hardware and software tuning**
  - **Adjusting thresholds ⟸ Price in Physics**
  - **New faster algorithms**
  - **Tracker design to aid track pattern recognition**
- As an example of the first two bullets: O(10) speedup achieved over the past 6 years in CMS reconstruction time using same PU~30 events, from a release mainly prepared for low-PU to a release set up to handle PU~30



CMS Status: HSF wshp 2016

Better use of computing and software

- Can't discount basic software optimization
  - **avoid redundant computations**
  - **avoid slow searching over unordered data structures**

- Commodity CPUs increase in complexity and we need to maximize use of the chip resources
  - **vector units, vector operations on large registers**
    - **e.g. a 256 bit register can manipulate 8 floats: use it**
  - **Many core architectures (Intel Xeon Phi, KNL etc)**
    - **e.g., arxiv.org:1409.8213,1605.05508 for KF-based tracking**
  - **GPUs can be put to use as well**
    - **e.g., ALICE upgrade HLT tracking**

Adjusting thresholds

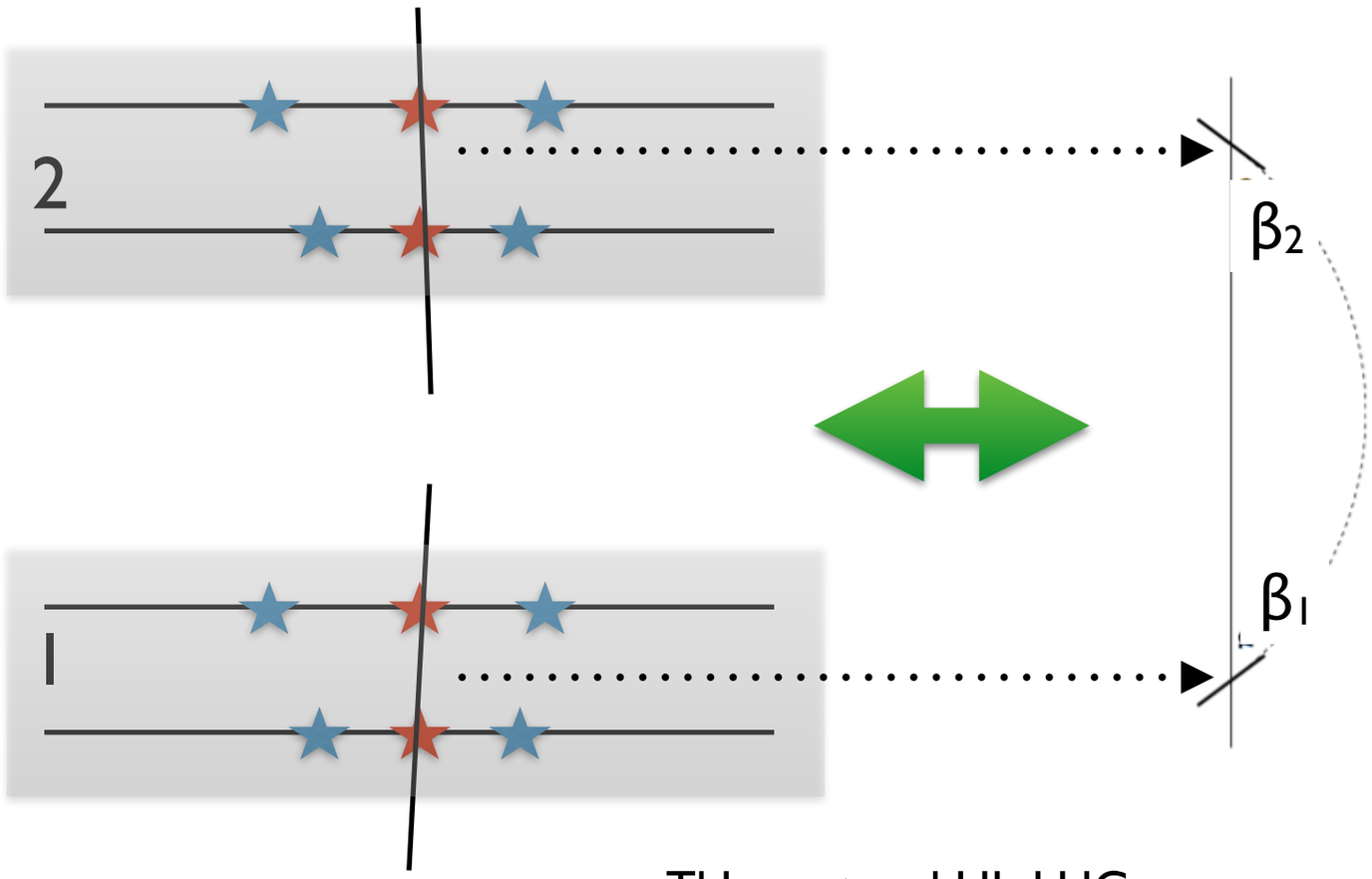
- We can achieve substantial reduction in CPU usage (and storage too!) by simply limiting the scope for the kinds of tracks we attempt to reconstruct
  - **increase pt threshold**
    - lose soft tracks
  - **limit the transverse distance to the beam line**
    - lose heavy flavor, cascades, secondaries
  - **limit distance of track origin to the beam line**
    - lose secondaries, conversions, long-lived particle decays
  - **consider only leading primary vertex (identified with very fast algorithms)**
    - lose good knowledge of other vertices
- All of these changes obviously imply a loss in physics performance of the experiment

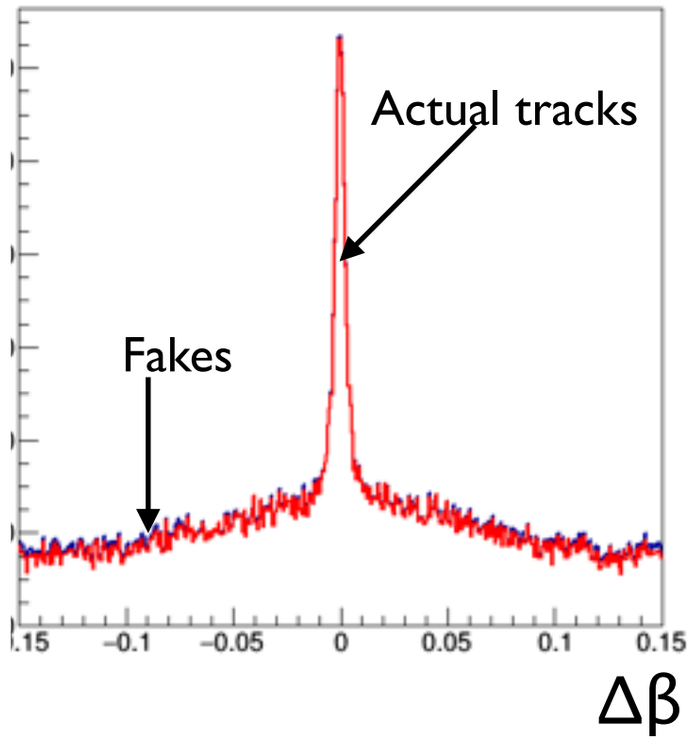
New faster algorithms

- NB: tracking is naturally divided in two parts 1) pattern recognition; 2) track fit.
- Pattern recognition: Connecting the dots across tracker layers doesn't have to be done with complex matrix operations
  - **Less mathematically optimal algorithms can be used to do tracking pattern recognition**
- Track final fit: still to be done with advanced fitting
- Variety of ideas and implementations
  - **Cellular automata approach**
  - **Precomputed pattern matching, most effective with associative memory arrays**
  - **Segment reconstruction and linking**
- These examples are not using progressive (sequential, hit-to-hit) global pattern building like the KF tracking. Instead, pattern elements are built locally from short hit sets (segments, tracklets) which are then combined with increasing hierarchy to create final tracks.
  - **Naturally, this is more applicable to many/multi-core CPU or GPU architectures**

Segment building and linking

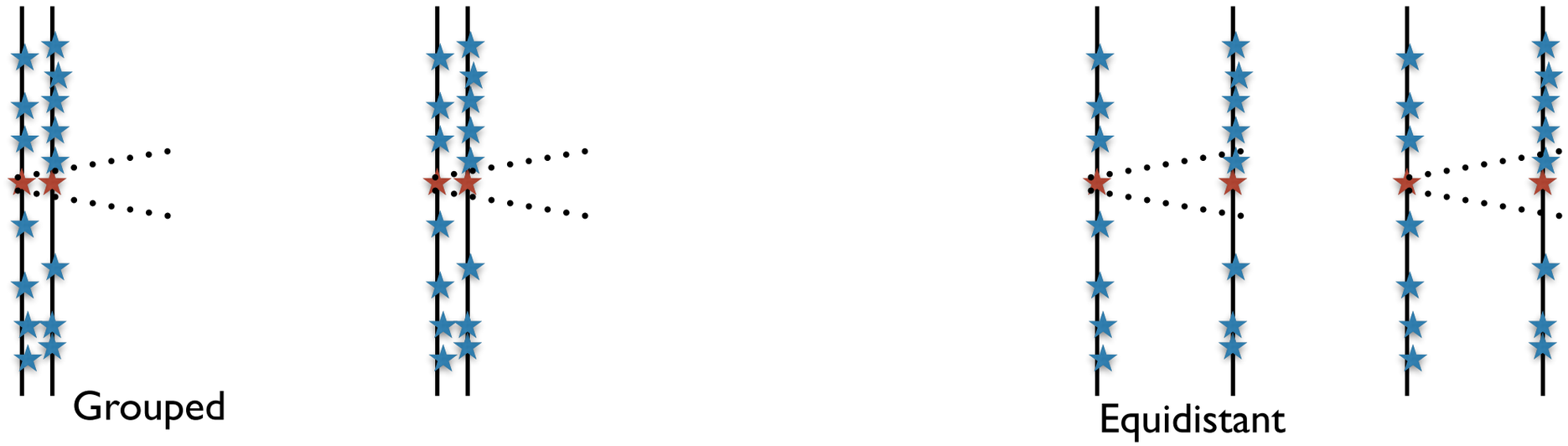
- An example fast algorithm
  - ✳ **Similar to what was used for the CDF drift chamber in '90s, developed by A. Mukerjee**
- Build: start with 2 layer groups (each can be multi-layer; need 2 layers to get direction) and reconstruct line segments in each group.
- Link: pick segment pairs consistent with a circle in xy (transverse to B-field) and with a line in r-z
  - ◉ **Correct combinations appear in the peak of Δβ distribution**
    - ✳ **width limited by material effects, detector resolution, pT, and precision of the algorithm**





Tracker layout to aid tracking: idea

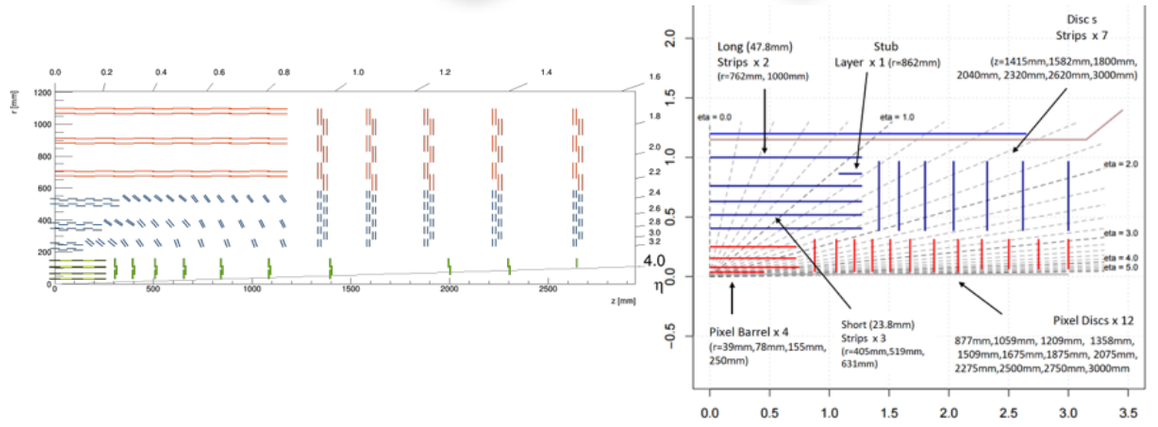
- … using segment linking
- Grouping layers to reasonably closely-spaced pairs should help with combinatorics from first principles



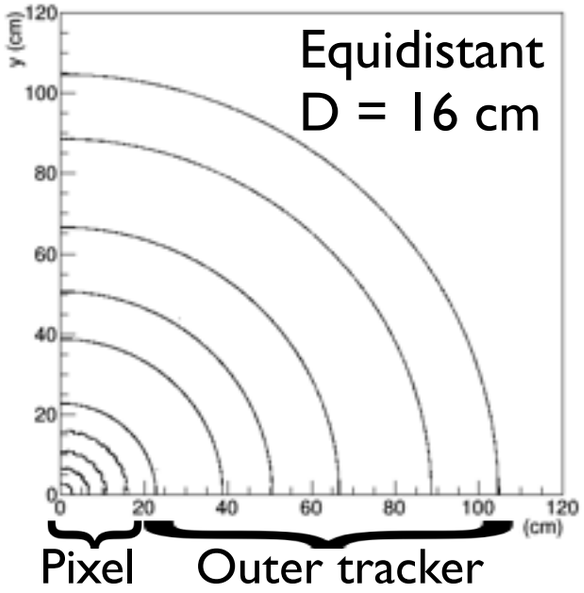
- The larger the layer separation, the larger area is swept for possible segment slopes to be considered giving fake segments.
  - **In the "grouped" and "equidistant" example**
    - **Segment direction is good enough in both cases to point from good segment to another**
    - **… but the same angle to consider to build possible segments leads to more combinations in equidistant case due to random hits**
  - **⇒ equidistant becomes increasingly costly CPU wise, especially as lower pT tracks are desired to be reconstructed**

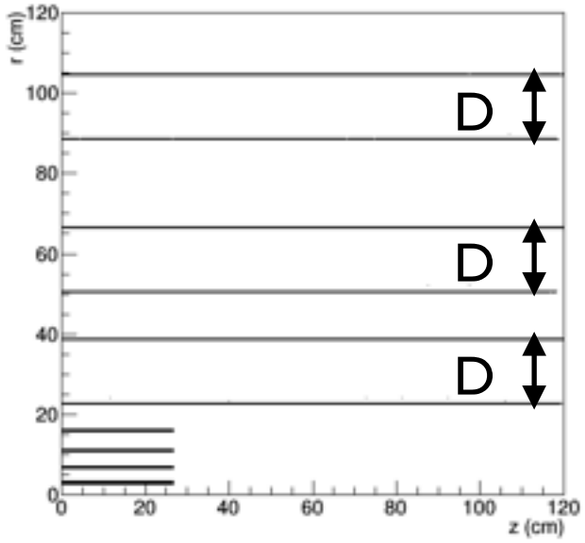
Tracker layout to aid tracking: setup

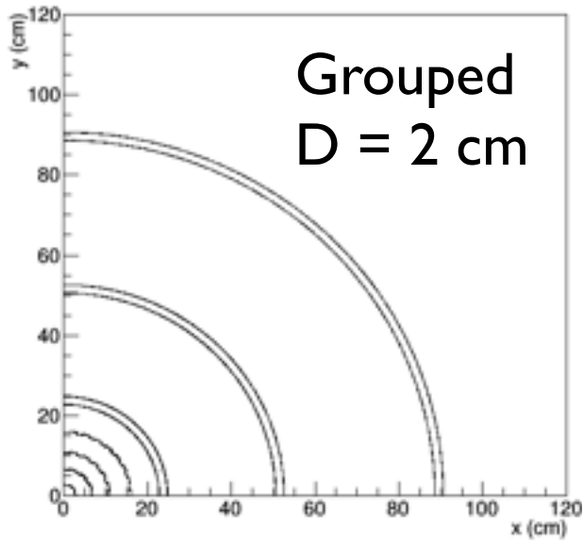
- Consider simplified design of the outer tracker for this study. Inspired by CMS/ATLAS layout.
  - ✳ **Inner precision pixel detector provides initial seeding for tracking**

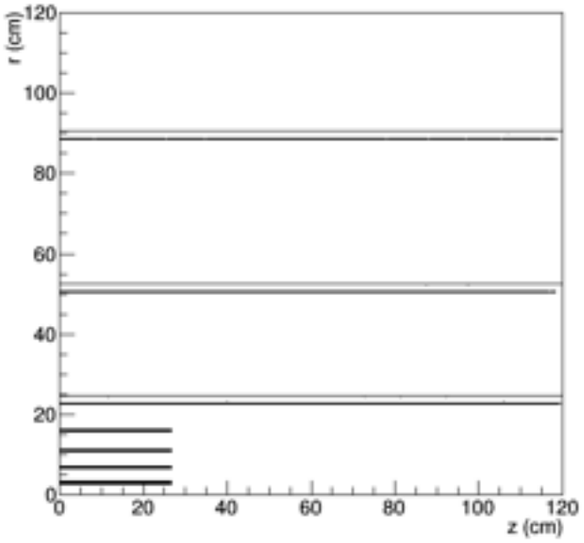


- Here, we explore possibility and gains from adjusting the layout to see impact on a faster pattern recognition, using segment linking. Consider only barrel.
  - ✳ **Mocked up geometry of outer tracker: 6 layers, fixed reference layer (5, 7, 9), vary separation for other layers**
    - ◉ **Equidistant (almost): use group distance D = 16 cm in the mock setup**
      - ✳ **Works well for hit-to-hit sequential tracking logic**
    - ◉ **Grouped layers: D = 2 cm in the mock setup**









Impact of tracker layer layout

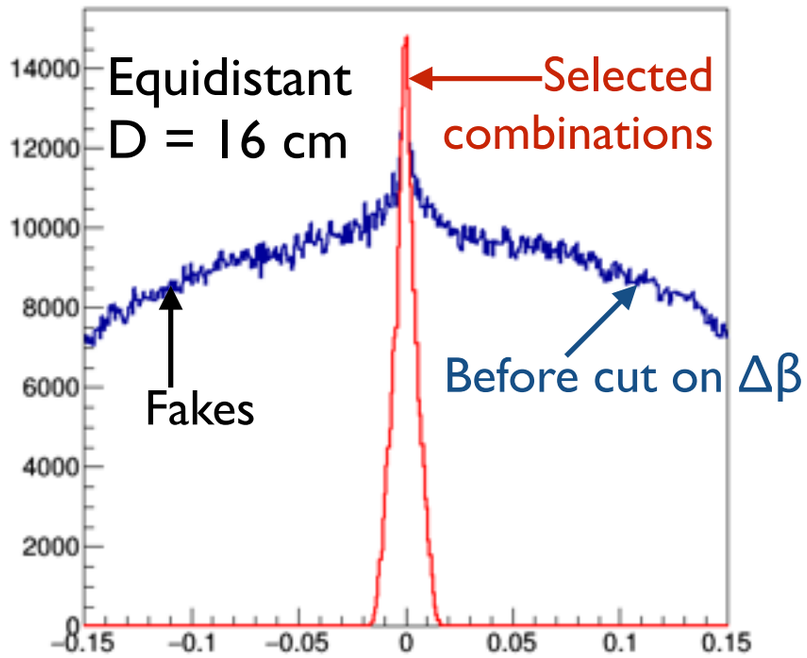
- Two relevant cases:

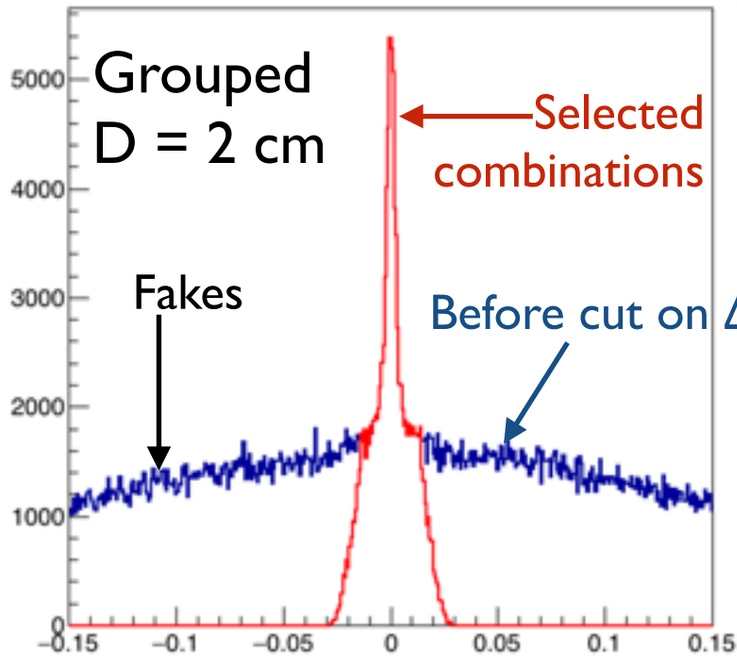
- Linking of inner pixel-based seed to a line segment in the outer tracker
  - **Inside-out tracking**
  - **Very pure and fast**
  - **Limited to prompt tracks**
- Linking two line segments in different layers of the outer tracker
  - **Becomes essential for tracking outside the pixel detector or cases missed by pixel seeding**
  - **Dominates resources usage**

- Take the second case to show impact of the tracker layout

Impact of tracker layer layout

- Linking two line segments: case essential for displaced tracks
  - ✳ **Figure of merit: size of combinatorial fakes (sidebands in Δβ), which correspond to CPU waste**





Plots for pileup 140,
with segment selections:
p_T > 1 GeV
in the barrel (|η| ≲1.6)

- Grouped choice is better: combinatorial fakes reduce substantially (by about 6)
- Overall CPU cost gain is large:
  - ✳ **Larger contribution from fakes in equidistant layout**
  - ✳ **Combinatorics dominates the total. Reduce combinatorics ⇒ reduce the total at similar rate**
  - ✳ **Multiple fits for each track candidate in the KF algorithm exacerbates the problem. Compared to ~one fit only in the segment linking approach**

Summary and outlook

- High luminosity high pileup operations of detectors are computationally challenging.
- Tracking carries a large fraction, using now conventional Kalman-based tracking pattern recognition.
- Faster pattern recognition algorithms can replace the conventional KF.

- Tracking detector costs and track reconstruction costs became comparable.
- Synergy between detector design and tracking algorithm decisions can help to reduce the total cost.

- We compare equidistant-layers tracker layout with grouped-layers tracker layout, using segment-linking algorithm as an example.
  - **We note that continued usage of traditional Patter Recognition approach is becoming un-affordable.**
- Grouped layer layout is expected to reduce computing costs, with substantial impact for tracking outside the (inner) precision pixel detector.

Random backup follows

Matching steps in segment linking

- 0: lumi region pointing
  - connect points on the inner and outer super-doublet (SD) layer and see it points back to luminous region (+/-15 cm)
- 1: Z pointing
  - For seed (L0) and SD from L5 and L7 use the inner SD direction to propagate and point to the z position on outer layer to end up on the +/-1 macro-pixel (3 mm) or strip (for L9 outer) range
    - Seeds: momentum direction for central value; for the matching width use eta-error and z-error of the seed as well as multiple scattering uncertainty (effect of fixed X=0.2, depends on seed pt)
    - SD: use two-point line in r-z from the inner as a direction, direction uncertainty is based on pixel size in z (resolution) and on multiple scattering for an assumed 1 GeV
- 2: inner-outer link slope
  - Slope of line connecting inner and outer SDs to be compatible (less than) slope of assumed pt: for seeds use the seed pt- 10sigma*ptError
- 3 (4): inner (outer) SD direction compatible with slope in 2.
  - For SD use 1 GeV for match range. For seeds use the seed pt (overlaps with dBeta).
- 5: deltaBeta cut
  - connect inner and outer SD to make large chord and compute beta angles with it and each SD (small chords). Cut on deltaBeta. For seeds, use pt to adjust small chord slope diff with tangent to the circle of the track in xy [seed beta is defined tangential to the circle]