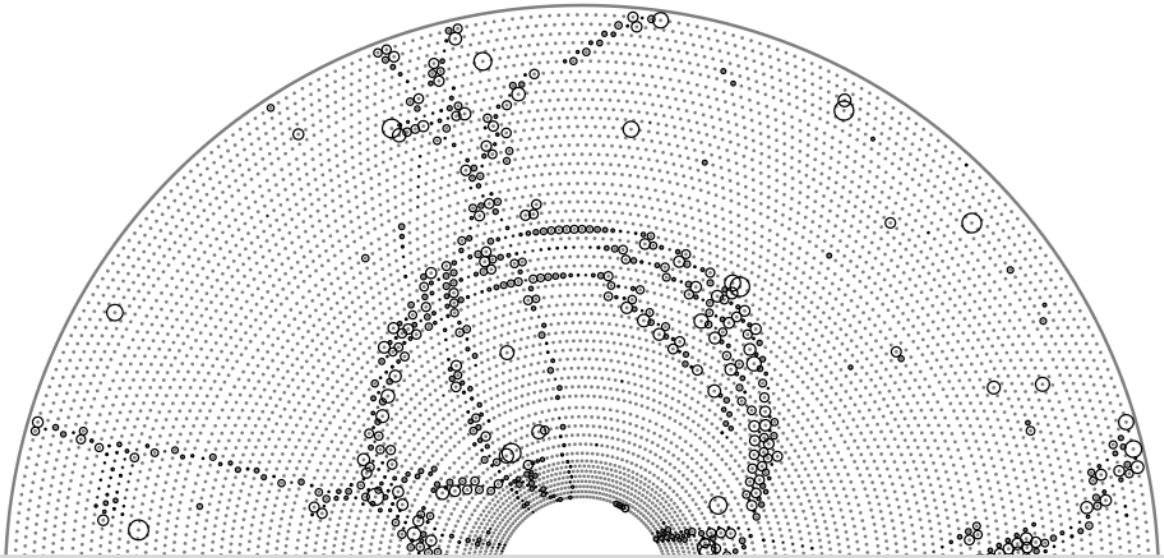
TrackFinderCDCLegendre: current status and new features

Viktor Trusov
04.21.2015 | F2F Meeting in Vienna

Karlsruhe Institute of Technology (KIT)



Quad Tree class template

- In Global track finder Quad Tree is the core of the algorithm

- Templating allows to use it not only for global track finding and solve such tasks as
  - Segment combination (see Nils' talk)
  - Apply for stereohits assignment
  - Use different algorithms for 2-d binary search

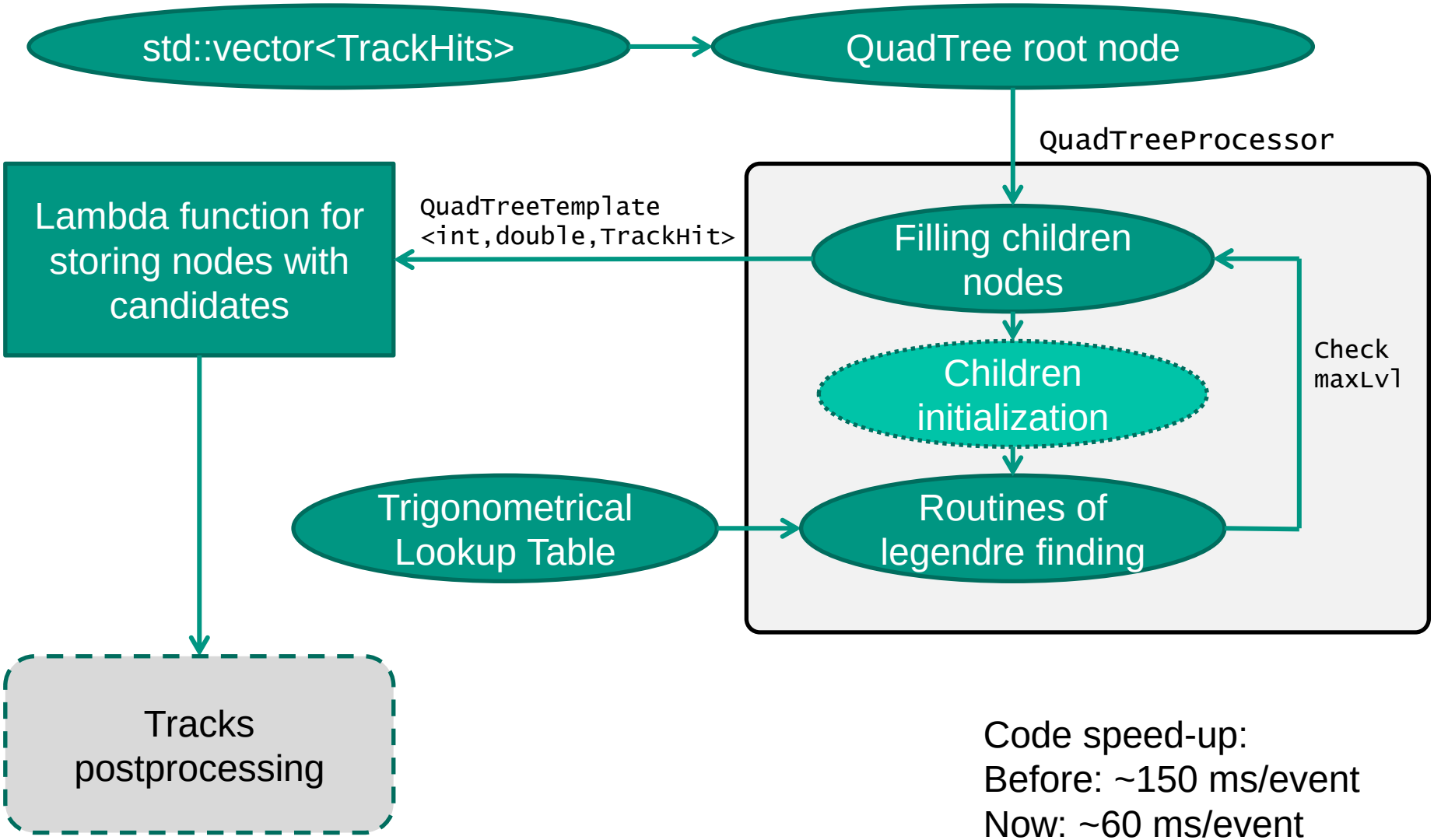
Quad Tree class template

- `QuadTreeTemplate<typeX, typeY, ItemType>` acts as dataholder:
  - Contains vector of items of type `ItemType`
  - Holds ranges assigned to current node
  - Holds pointers to parent and children nodes

- For working with `QuadTreeTemplate<>` structure processor class should be implemented. Generally must contain:
  - Routines for creating children nodes
  - Routines for filling nodes

- Templated Quad Tree can be applied for 2-d binary search in any phase space
  - But correct routines for working with objects of class `ItemType` should be implemented

How it works



Code speed-up:
Before: ~150 ms/event
Now: ~60 ms/event

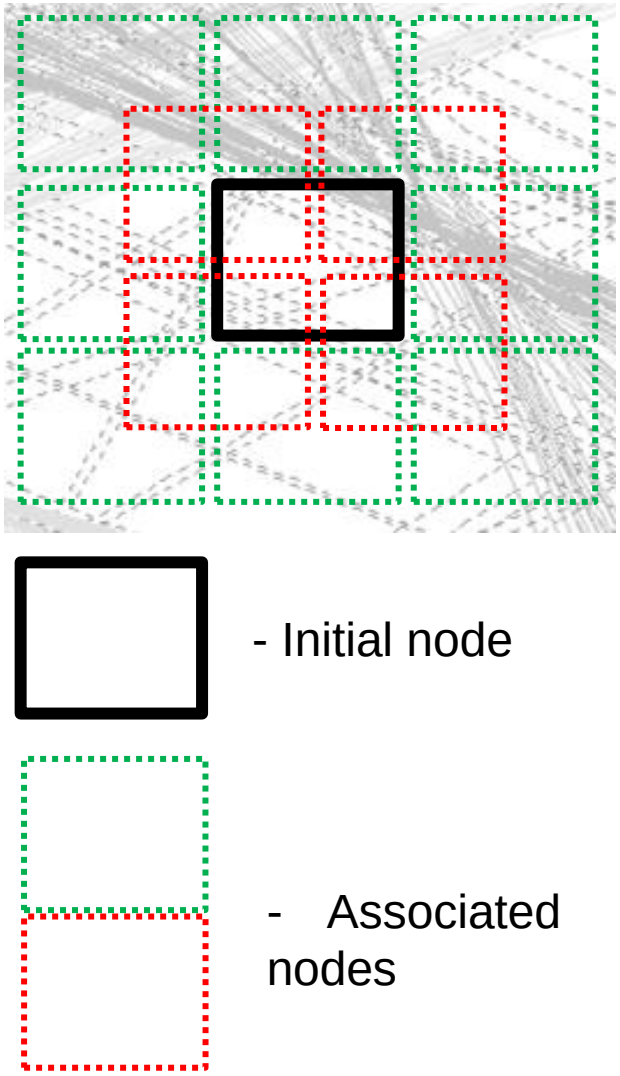
New routines for candidate postprocessing: Transformation with respect to given point

- This routine allows to add more hits to the track using circular fit and single quad tree node
- After fitting track POCA is taken as new reference point
- single QuadTree node is created and filled
  - Boundaries of node defined using parameters of the candidate
    - as $r_0 \pm \Delta r;\ \theta_o \pm \Delta\theta$
    - r_0, θ_o calculated from fit parameters
    - $\Delta r, \Delta\theta$ estimated using p_t of track
  - Legendre transformation performed with respect to new reference point

- This procedure allows to increase hit efficiency
  - Eliminates effect of spread of hits in legendre phase-space

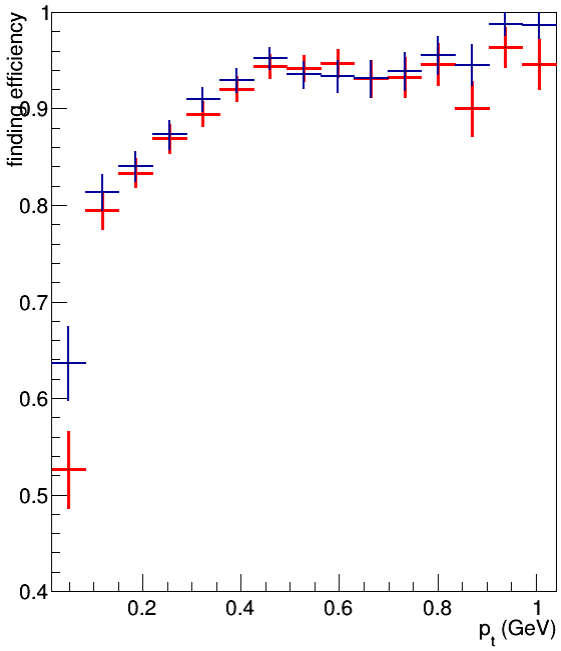
New routines for candidate postprocessing: building associated nodes

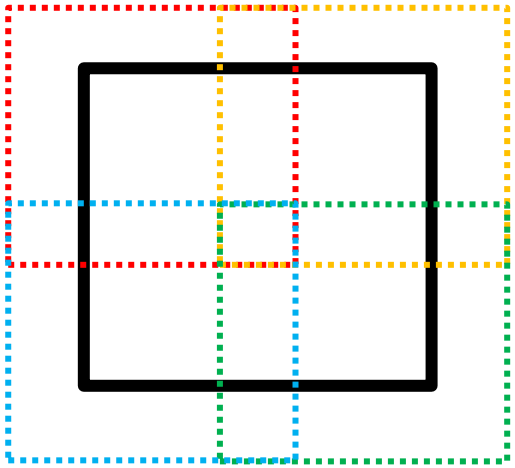
- Track could be broken if intersection point of hits in legendre phase space is near to borders of QuadTree bins
- To decrease probability of breaking tracks associated nodes are built and filled
  - 8 neighboring nodes
  - 4 overlapped nodes
- Hits from most populated node is added to the track



Low-pt region

- It turned out that for low-pt region deepness level of QuadTree should be much lower
- With decreasing maxLevel parameter for pt < ~150 MeV efficiency was increased
- Another effect – increasing boundaries of children





What we gain?

	fake_rate	finding_efficiency	clone_rate	hit_efficiency
build-2015-04-03	0.3068	0.9155	0.0471	0.5512
build-2015-03-01	0.3150	0.8795	0.0475	0.4999

New tool: LegendreResolutionTest

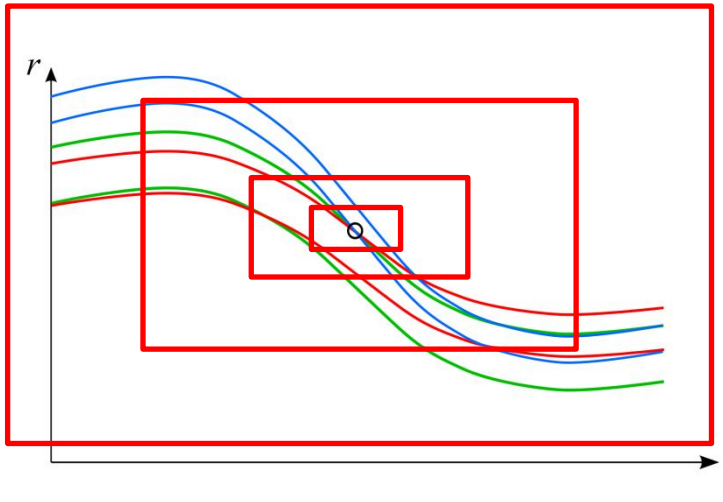
- Intersections of hits in legendre phase space are affected by spreads
- Spread depends on p_t and $d0$ of the track
  - Due to energy losses *low-pt* tracks are mostly affected by spread
  - With increasing *d0* spread also increases

- We are able to tune QuadTree search strategy by introducing resolution which characterizes spread
- Main idea: make tool which allows to estimate best parameters of QuadTree
  - was: node marked as “with candidate” if reached maximal level of QuadTree search
  - should be: node possibly contains candidate if it’s “size” smaller than defined resolution

- In future could be applied to real data

How to measure spread and resolution?

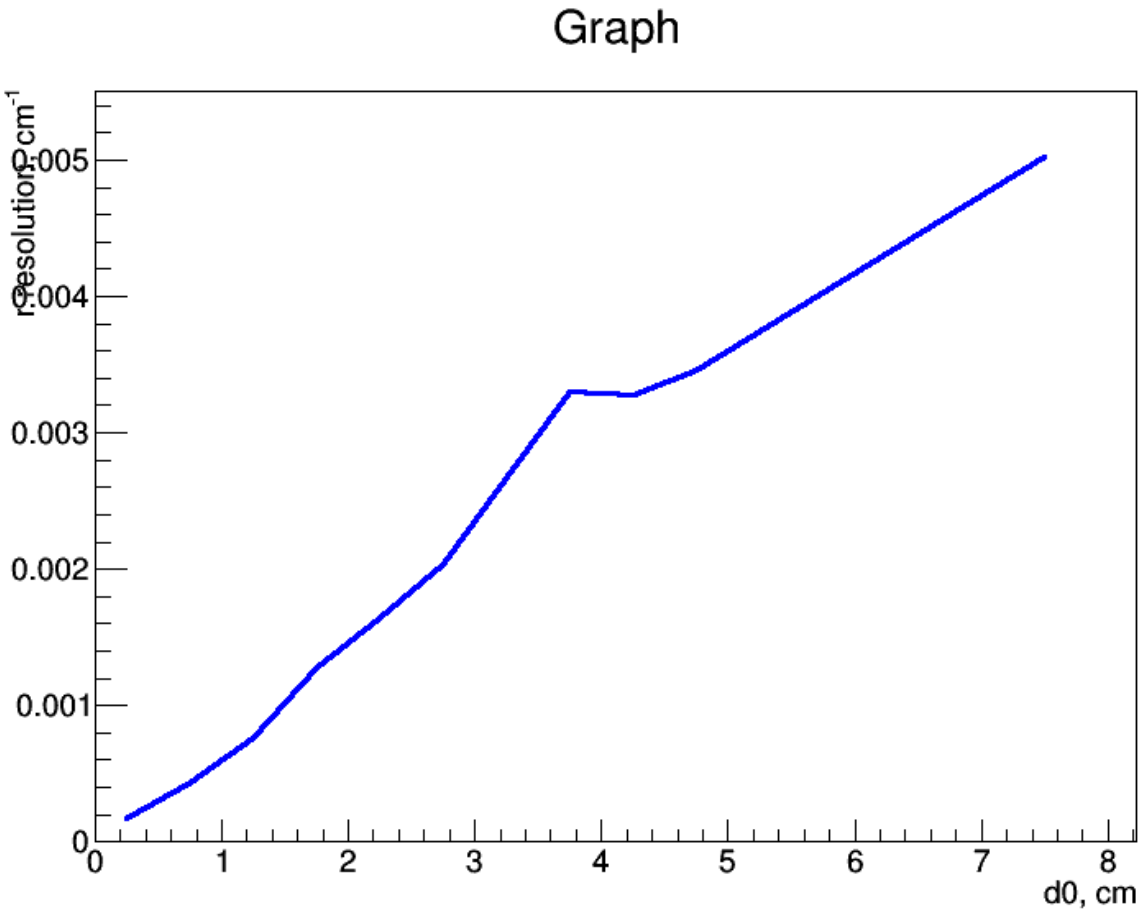
- Using particle gun single tracks were generated
  - $p_t < 2\ GeV$
  - $d0$ – normal distribution with $\sigma = 3\ cm$
- Using parameters of the track single QuadTree node was created
  - QuadTree node centered
- Boundaries of the node taken as whole legendre phase space
- If all generated hits could belong to the node – reduce its size by factor 2
- Repeat until desired number of hits still belongs to the node



- Final size of node charecterizes resolution for track with given parameters

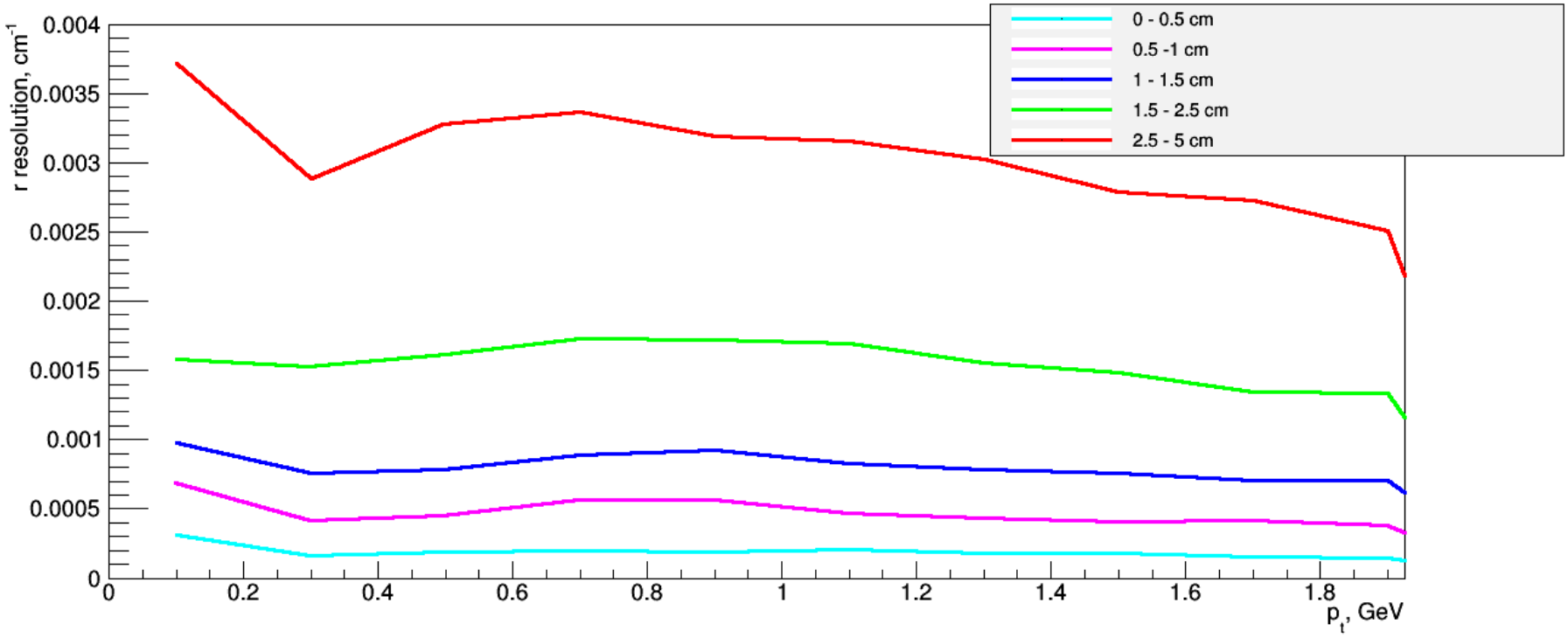
Resolution – *d0*

- 40 000 pions were generated
- Resolution strongly depends on $d0$

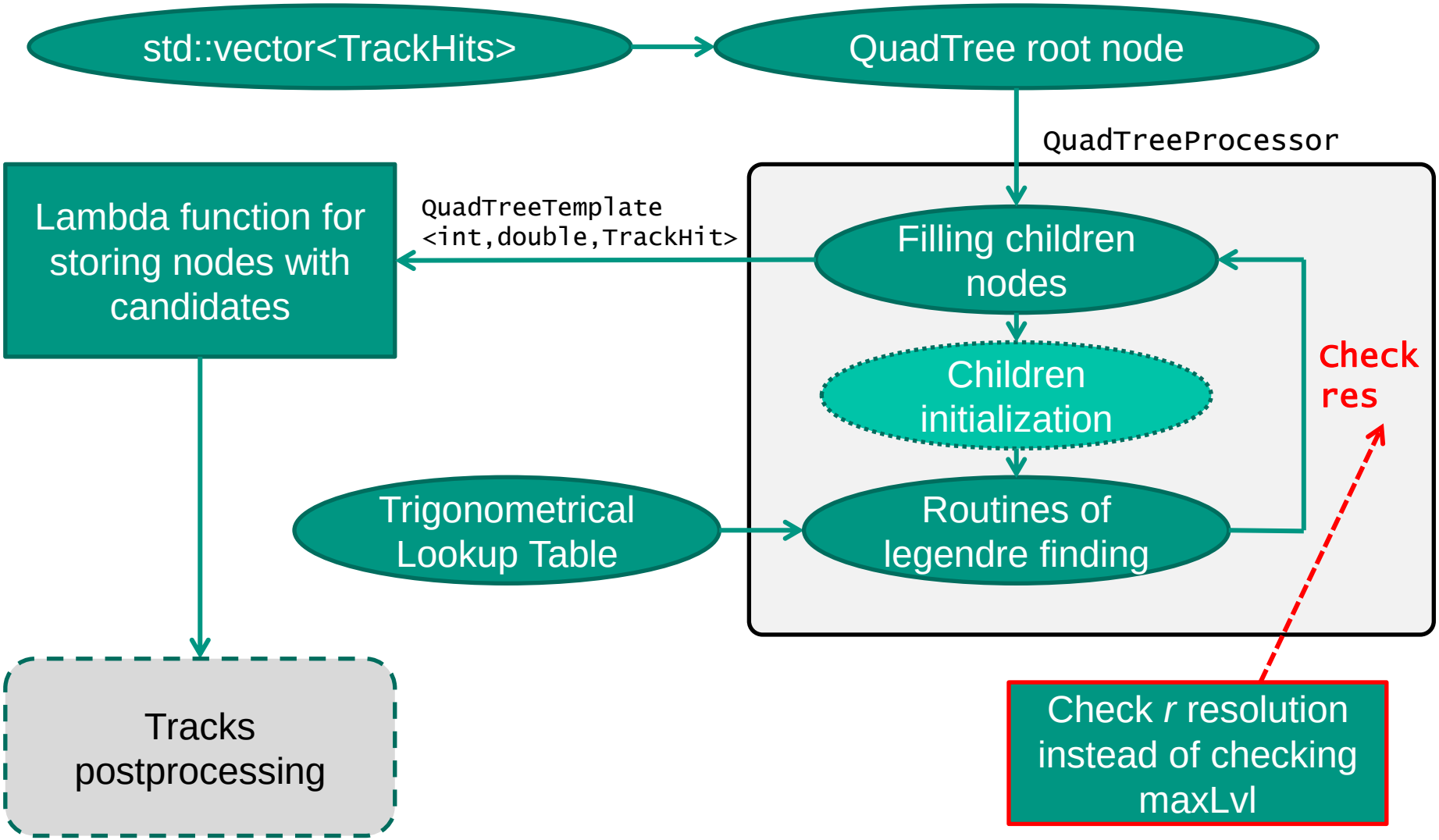


Resolution – *pt*

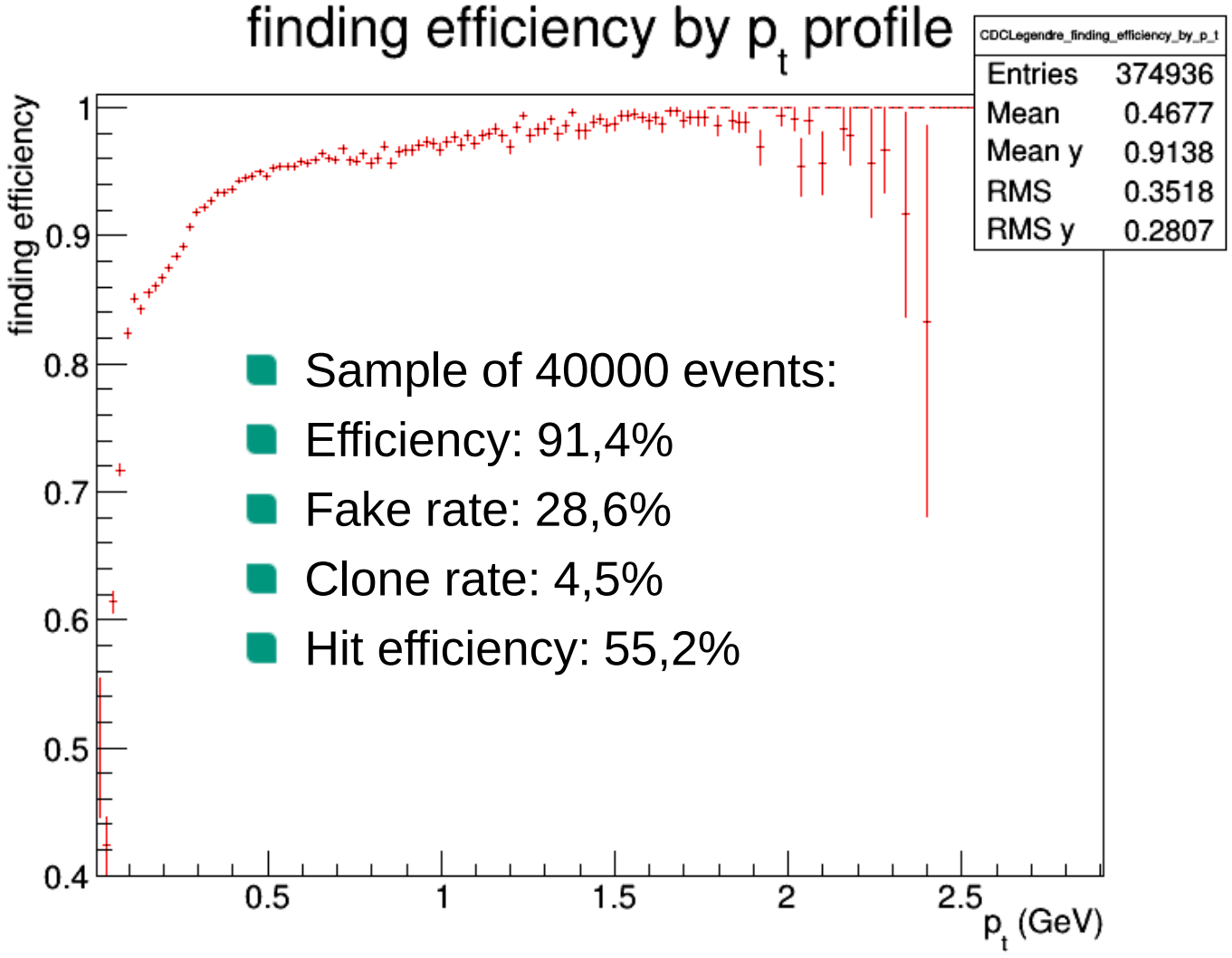
- Resolution also depends on p_t
  - For lower p_t it's higher (as expected)
  - Unexpected: bump around 0.8 GeV



How to apply it



Efficiency


finding efficiency by p_t profile
CDCLegendre_finding_efficiency_by_p_t
Entries 374936
Mean 0.4677
Mean y 0.9138
RMS 0.3518
RMS y 0.2807
finding efficiency
1
0.9
0.8
0.7
0.6
0.5
0.4
0.5
1
1.5
2
2.5
p_t (GeV)
Sample of 40000 events:
Efficiency: 91,4%
Fake rate: 28,6%
Clone rate: 4,5%
Hit efficiency: 55,2%

Conclusions

- Few features were presented
  - Increased quality of track finding
- New tool in development
  - Will help to tune Global track finder
  - May be applied to real data

Thank you for attention!