SLHC Tracking Trigger Framework and Utilities

Andy Rose

Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

A little terminology

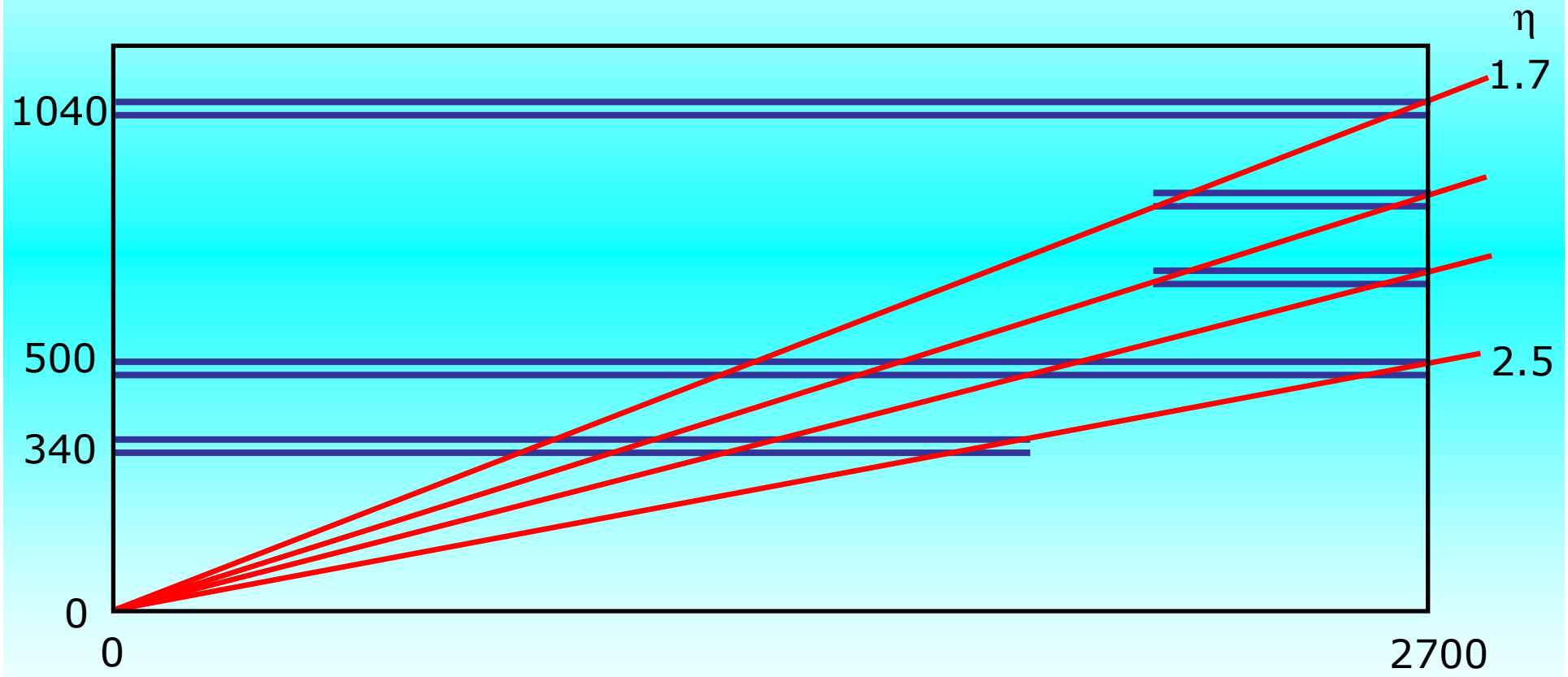
The following definitions have been agreed upon by a panel of members from the tracker upgrade and track-trigger groups.

- two or more sensors seperated by O(mm) = a stack
- the sensors within a stack = stack members.
- a high pT correlation between hits in a stack = a stub
- two, three, four, ... stacks separated by O(cm)→O(m) = a double, treble, quadruple, ... stack
- a correlation between stubs in a double, treble, quadruple, ... stack = a tracklet
- the basic element from which trigger primitives are produced = a station (be that a stack, a double stack, a treble stack,...)

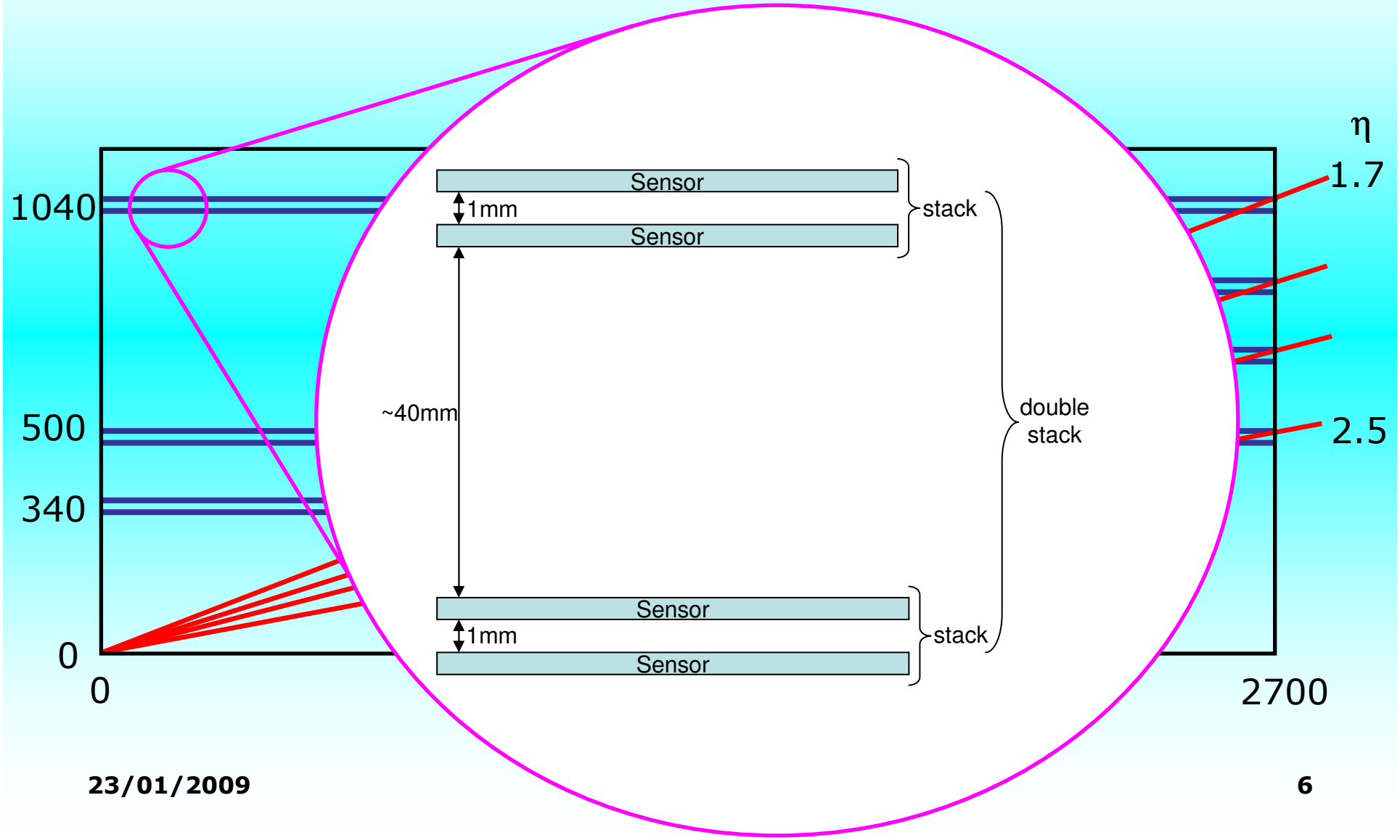
The terms **doublet**, **superlayer** and **superstack** have all been depreciated and should not be used.

The 'Fermilab' Geometry

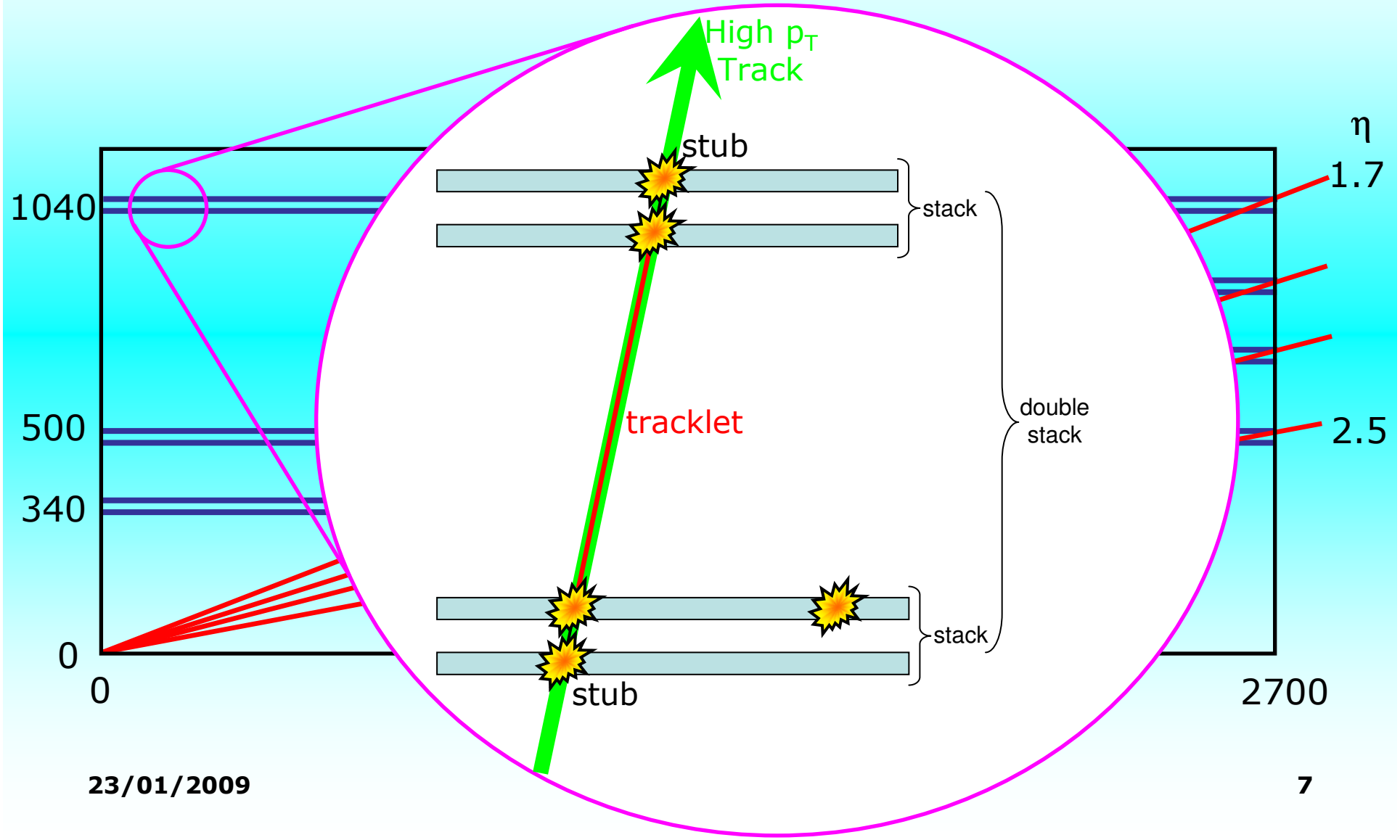
'Baseline' upgrade design agreed at
Fermilab workshop 24th November 2008



Looking a little closer…



Looking a little closer (2)…



Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

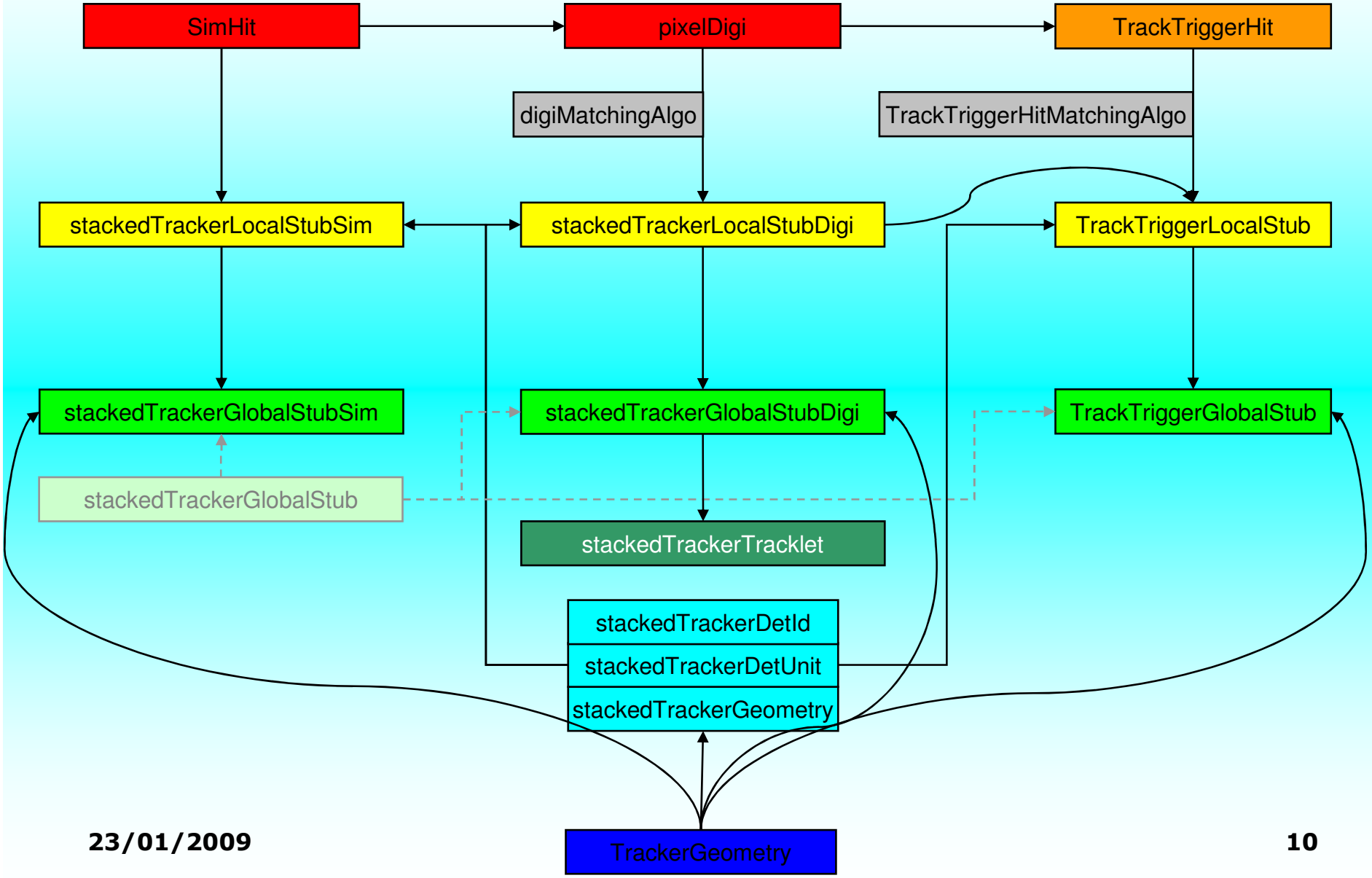
Simulation infrastructure

The strawman b family of upgrade geometries (of which the FermiLab geometry is one) do not describe stacks per se, but rather a collection of individual detector element positioned as pairs.

The current framework makes no provision for handling such objects – a new type of detector object is required.

The current framework also makes no provision for stubs or tracklets – new data formats are required.

Overview

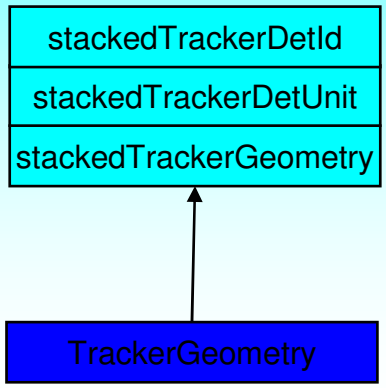

SimHit
pixelDigi
TrackTriggerHit
digiMatchingAlgo
TrackTriggerHitMatchingAlgo
stackedTrackerLocalStubSim
stackedTrackerLocalStubDigi
TrackTriggerLocalStub
stackedTrackerGlobalStubSim
stackedTrackerGlobalStubDigi
TrackTriggerGlobalStub
stackedTrackerGlobalStub
stackedTrackerTracklet
stackedTrackerDetId
stackedTrackerDetUnit
stackedTrackerGeometry
TrackerGeometry

Stacked Tracker geometry utilities

StackedTrackerDetId: A DetId class uniquely encoding the subdetector (barrel/ endcap), layer, iφ and iz.

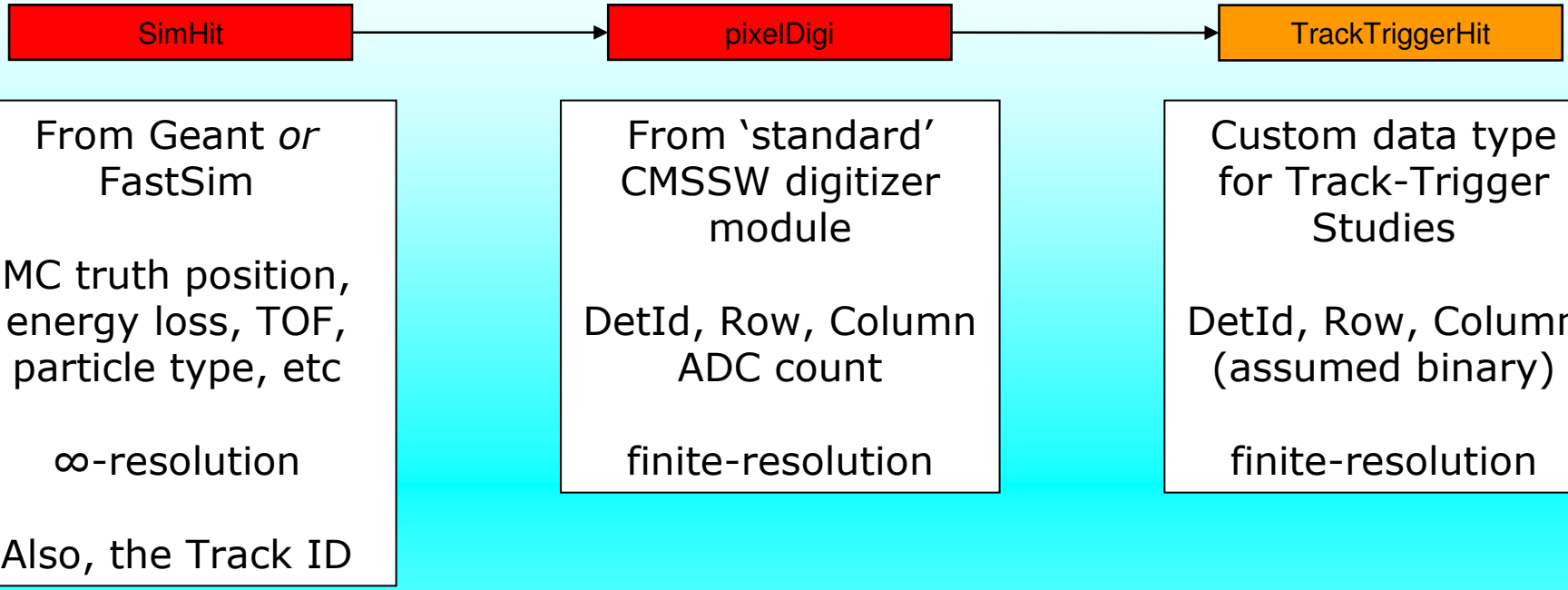
StackedTrackerDetUnit: An object containing a sorted list of the DetIds of the stack members and having a StackedTrackerDetId.

StackedTrackerGeometry: Contains a list of stackedTrackerDetUnits and also provides various helper methods for association of StackedTrackerDetIds to StackedTrackerDetUnits.



Stacked tracker can be treated like any other piece of hardware in the detector

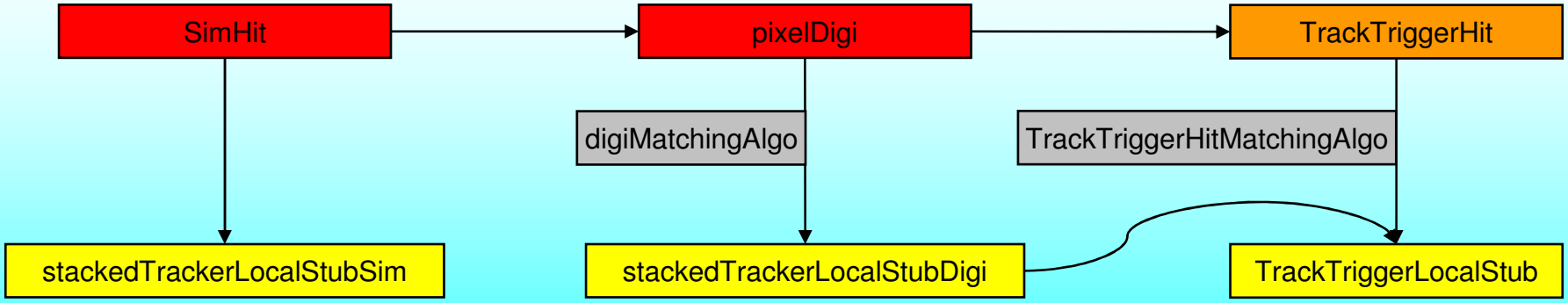
Stacked Tracker data formats (i)



For clarity, on the following slides discussion of the TrackTriggerHits format may be omitted.

Everything discussed for pixelDigis is equally applicable to TrackTriggerHits

Stacked Tracker data formats (ii)

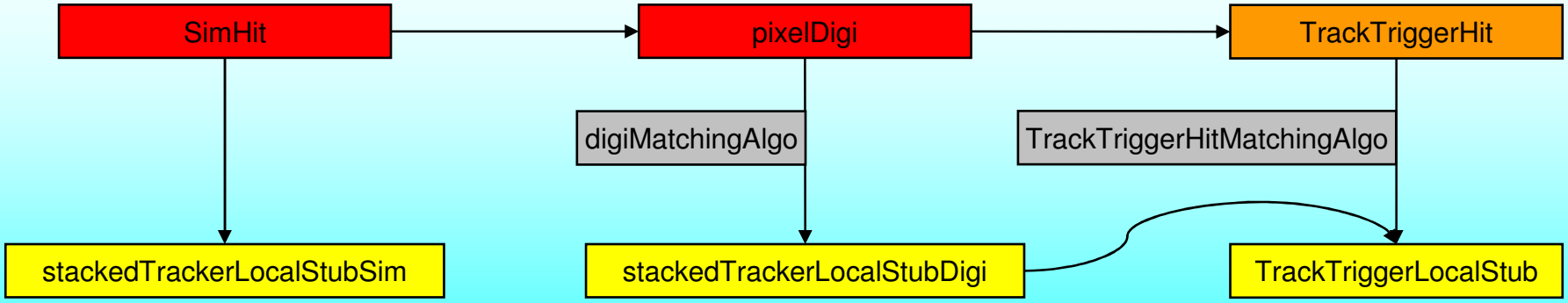


LocalStubs are sorted lists of hits† within an event, with an associated StackedTrackerDetId.

It is envisaged that this is the type of object that will be formed on-detector.

†technical note: the format is actually as persistent references to the framework hits

Stacked Tracker data formats (iii)



simHits are matched geometrically in the global frame

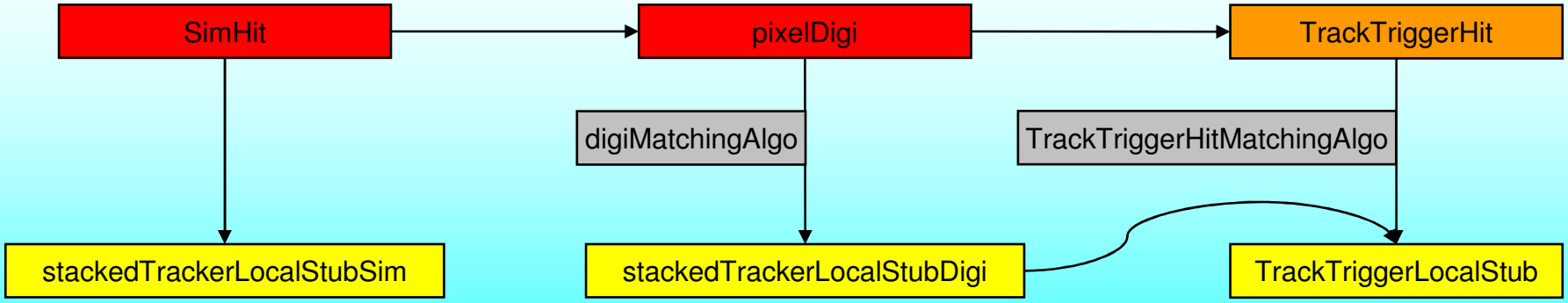
Cut on ΔΦ based on a p_T threshold

Cut on the projected vertex position

Both cuts set the config file

This algorithm is the "best case scenario" as simHits have infinite position resolution and using global geometry removes any geometric dependencies

Stacked Tracker data formats (vi)



The global geometry algorithm is again the "best case scenario" as it removes any geometric dependencies.

It is *NOT* implementable on-detector but gives a figure for comparison

pixelDigis (or TrackTriggerHits) are matched with a matching algorithm

The matching algorithm is a separate entity to the code which adds the stubs to the event and is included as an ESModule†

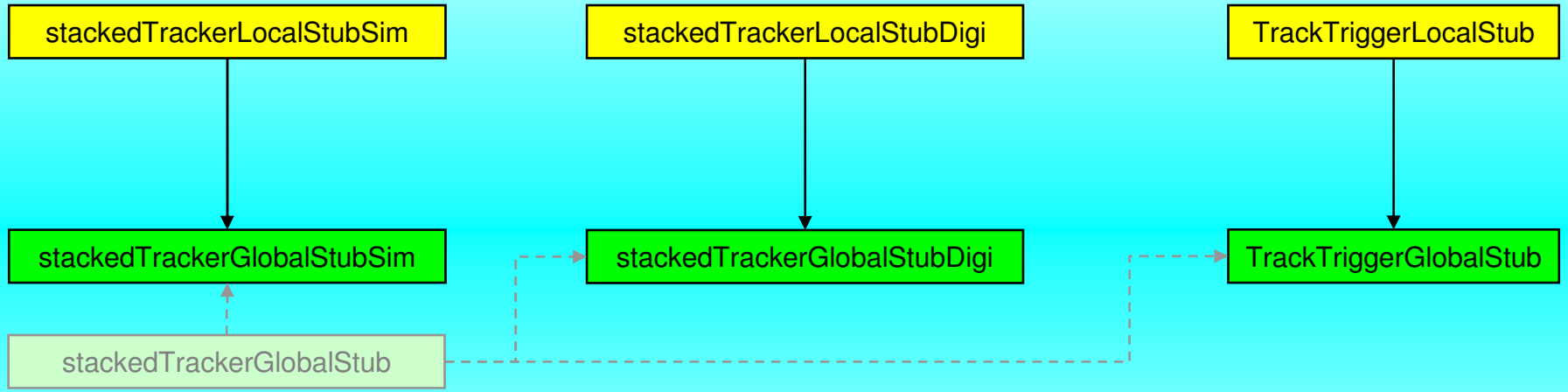
Current studies use a global geometry algorithm for direct comparison to stubs from simHits

Trivial to code an algorithm based on row/column windows (of course validation and optimization will require a little more work ☺)

†technical note: the choice of algorithm is made by using the relevant include statement in the config file

Stacked Tracker data formats (v)

GlobalStubs are geometric objects with a global position (the average global position of the two constituent hits) and global direction (the vector between the two constituent hits).



It is envisaged that this is the type of object that will be used in the level-1 trigger for association of hits between stacks.

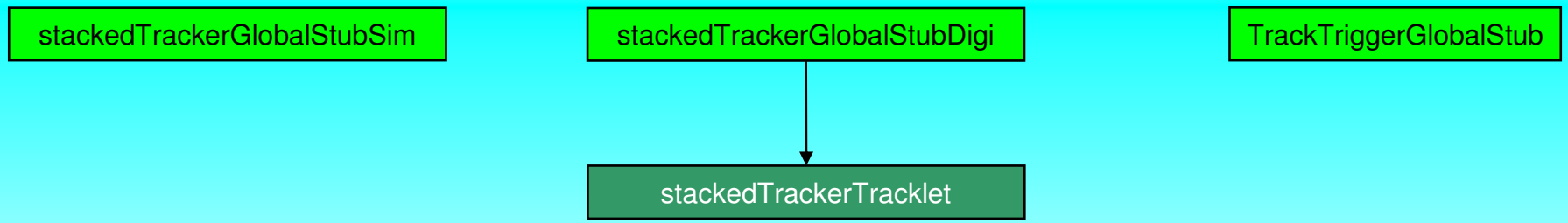
Currently they are produced one-for-one from the local stubs.

Stacked Tracker data formats (vi)

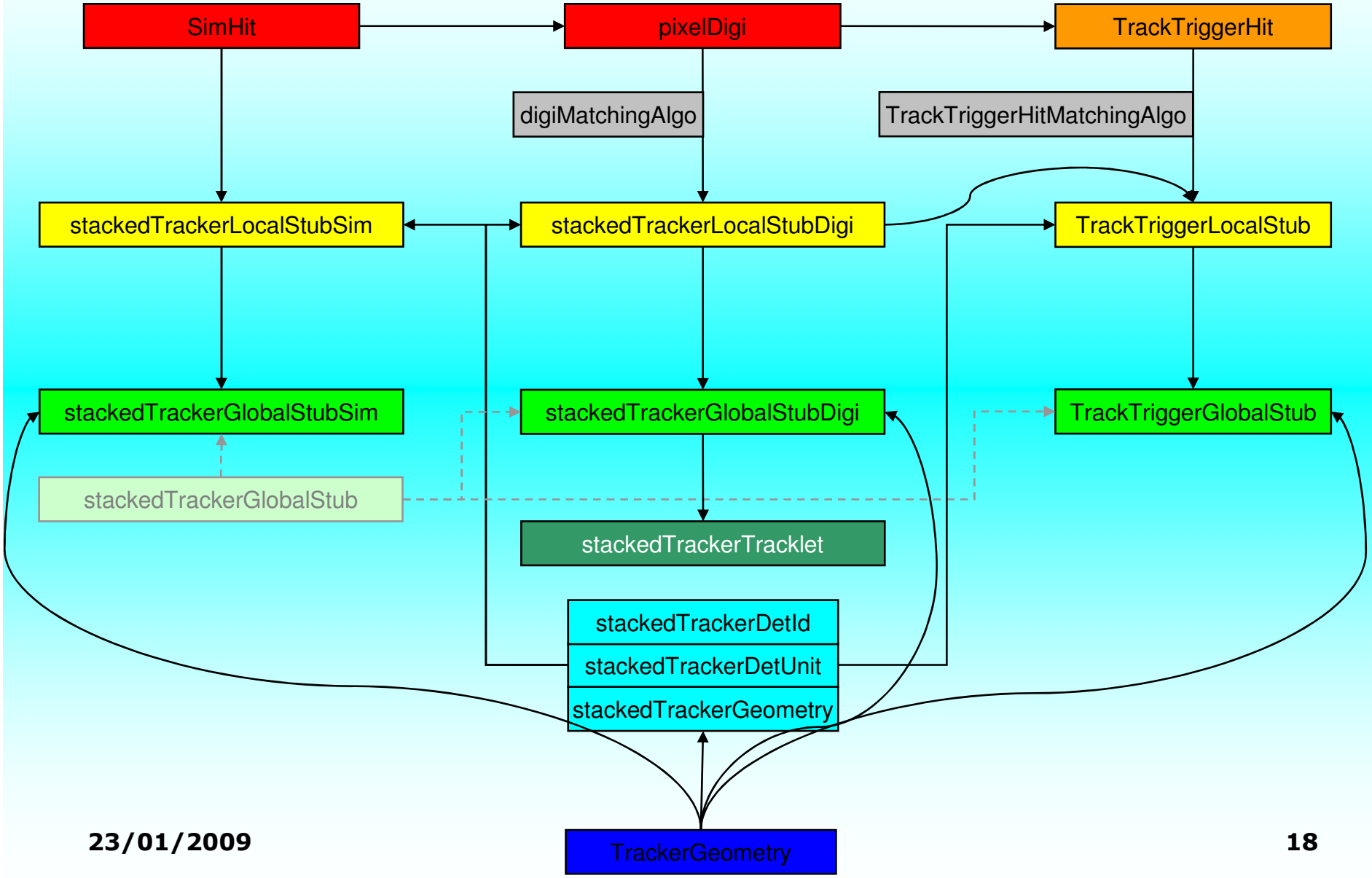
WORK IN PROGRESS!!!

Tracklets are formed geometrically from global stubs by placing a cut on ΔΦ based on a p_T threshold and a cut on the projected vertex position

Currently only performed between pairs of consecutive layers although framework allows for any number of stubs in a tracklet



Overview

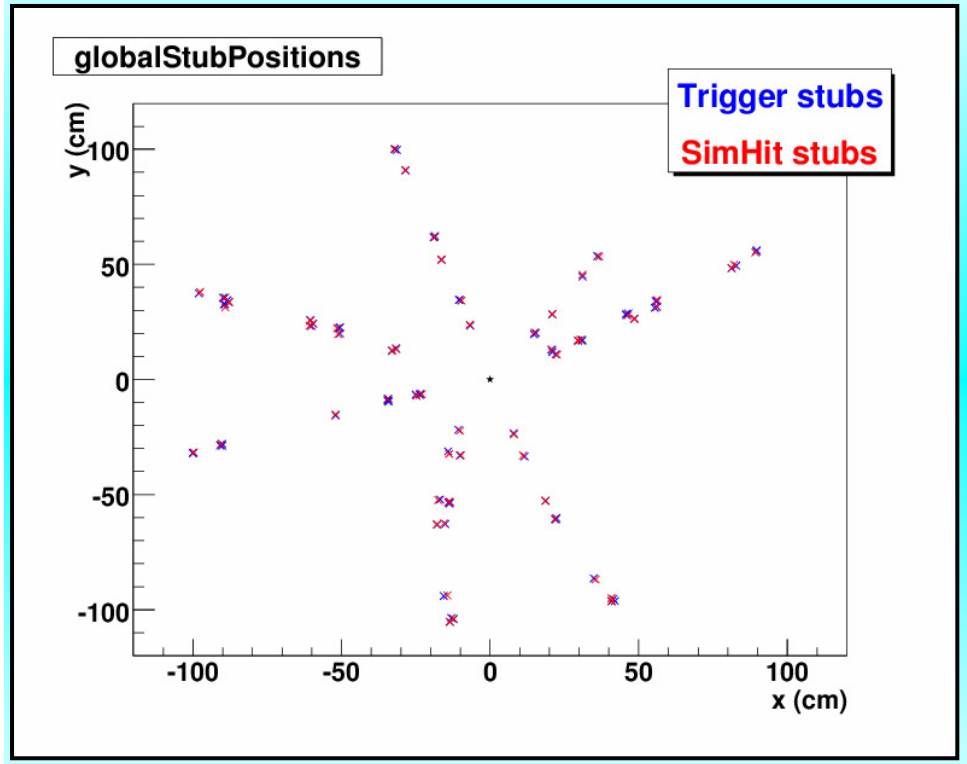

SimHit
pixelDigi
TrackTriggerHit
digiMatchingAlgo
TrackTriggerHitMatchingAlgo
stackedTrackerLocalStubSim
stackedTrackerLocalStubDigi
TrackTriggerLocalStub
stackedTrackerGlobalStubSim
stackedTrackerGlobalStubDigi
TrackTriggerGlobalStub
stackedTrackerGlobalStub
stackedTrackerTracklet
stackedTrackerDetId
stackedTrackerDetUnit
stackedTrackerGeometry
TrackerGeometry

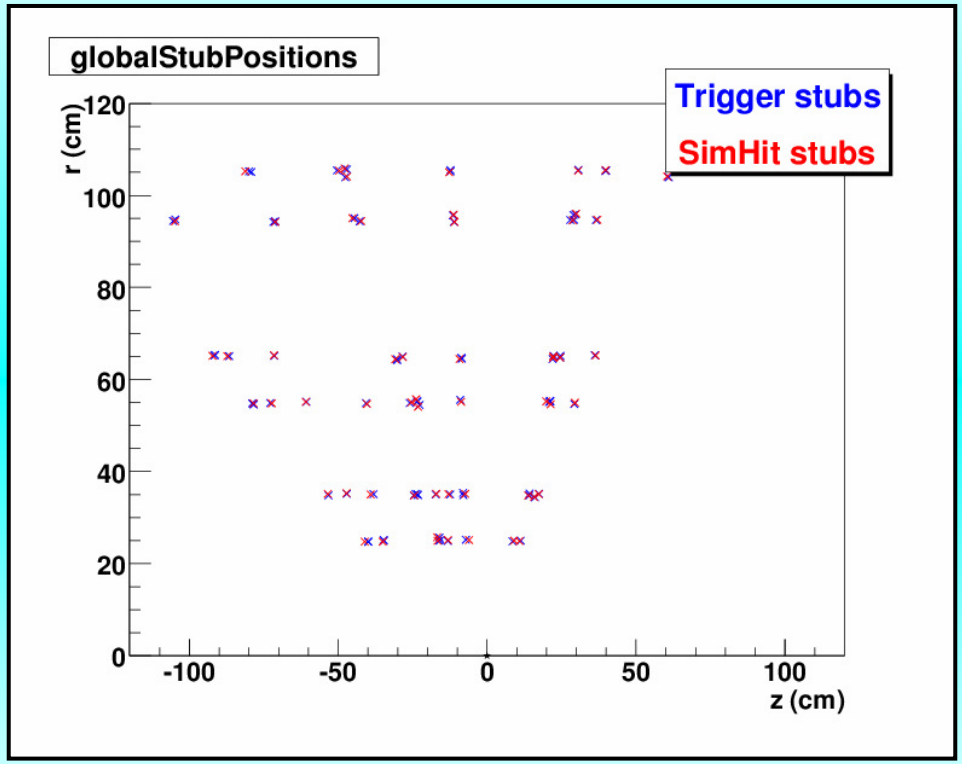
Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

Some results (i): Can we see stubs?

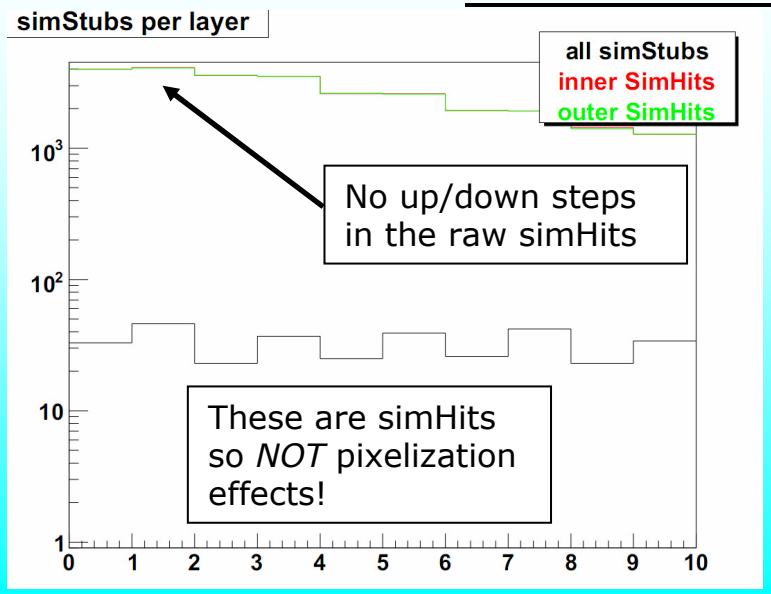
The software includes validation code for checking that results are sensible



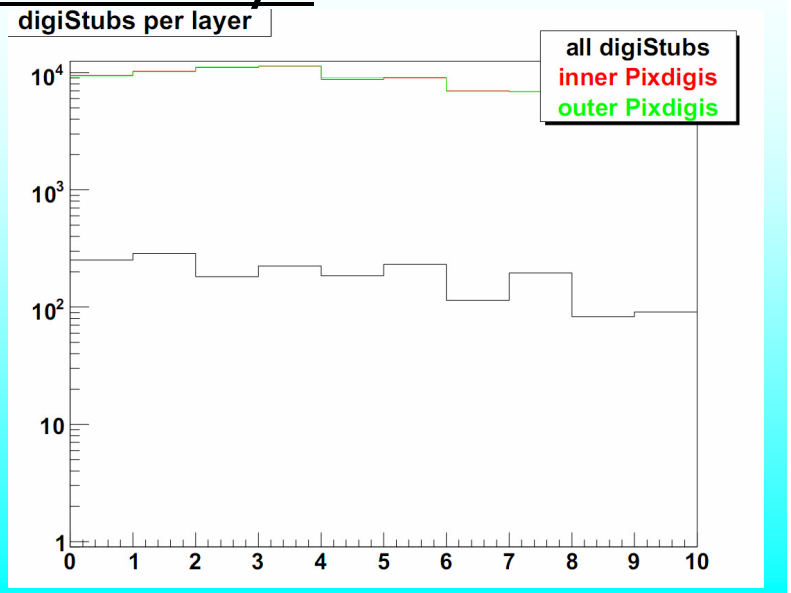


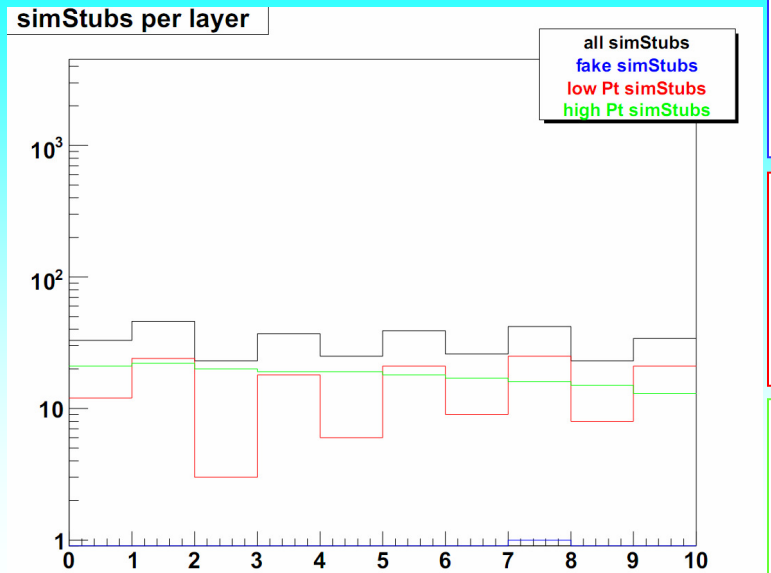
10 muon gun events (pT = 5→25GeV) with a pT cut of 1GeV and no cut on vertex Z

Some results (ii): How many?



100 crossings with 20PU in fast simulation

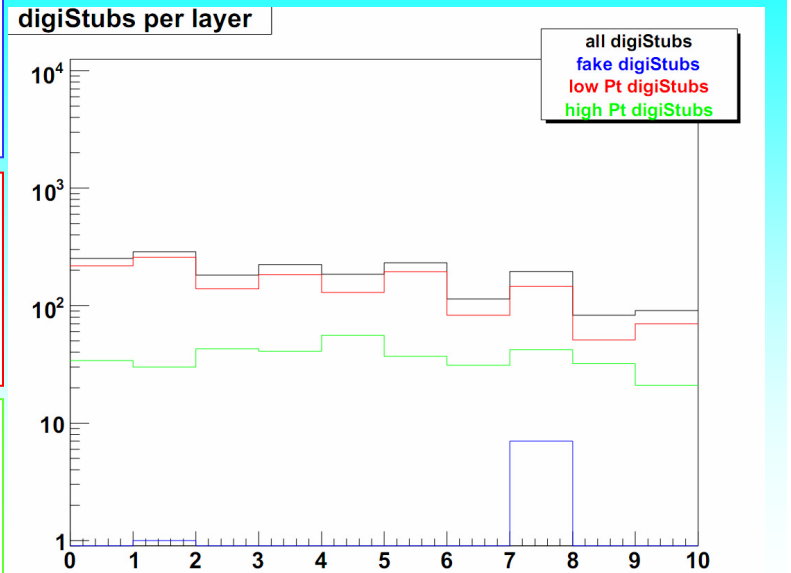




Those where the track Ids† of the two hits are different

Those where the track Ids† of the hits match and correspond to a track below 5GeV

Those where the track Ids† of the hits match and correspond to a track above 5GeV



†There are subtleties!

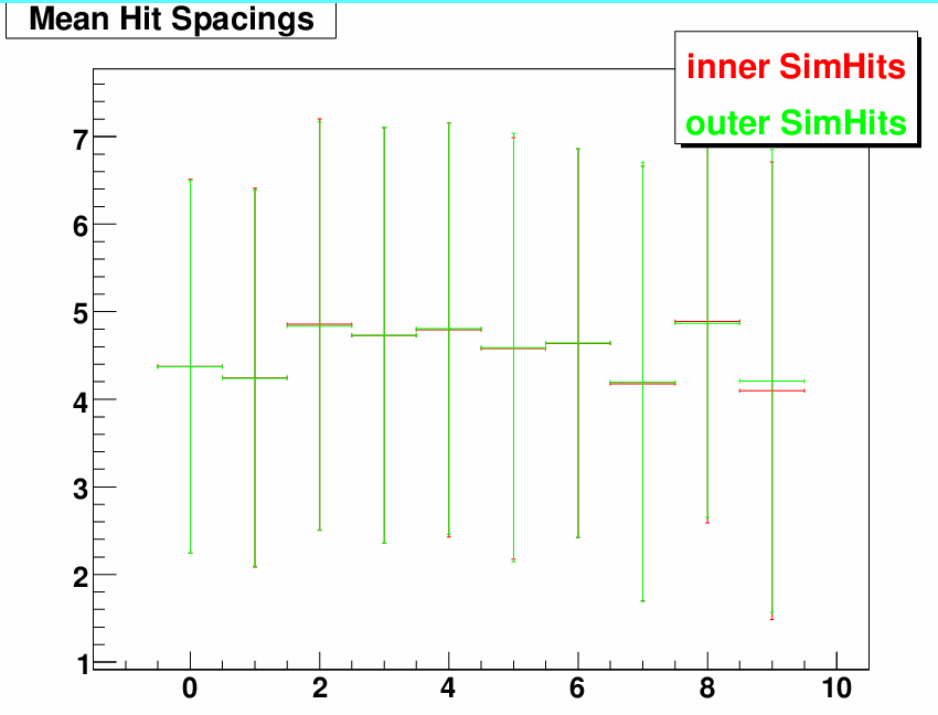
Some results (iii): How many?

Because the effect is not seen in the total number of hits in the layer this necessarily implies that the effect lies with the probability of stub formation.

The probability depends on the local density of hits in a stack member rather than on the total number of hits.

Plotting the spacing for all combinations of hits *within each sensor*† provides us with a crude measurement of how tightly clustered the hits are.

As expected from the plots on the previous slide, hits are more tightly clustered in odd-numbered layers leading to higher probability of stub formation!



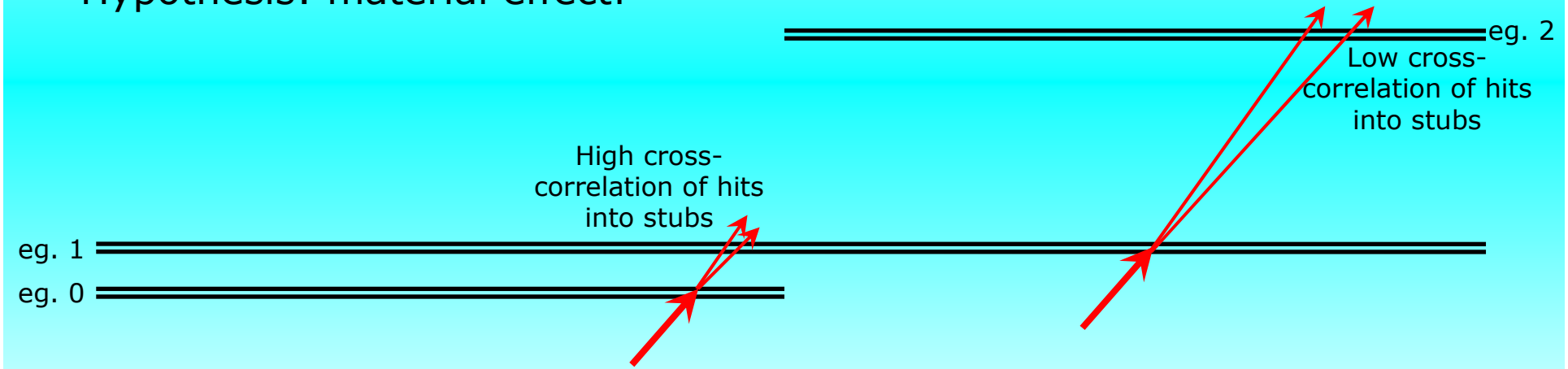
†Please note! This is not stub formation! This is just looking at the hits within one silicon sensor element!

Some results (iv): How many?

This is not seen in strawman-B… Why?

Strawman-B has relatively large spacing between stacks† whereas the FermiLab geometry has the odd-numbered layers positioned close behind the even-numbered layers.

Hypothesis: material effect!

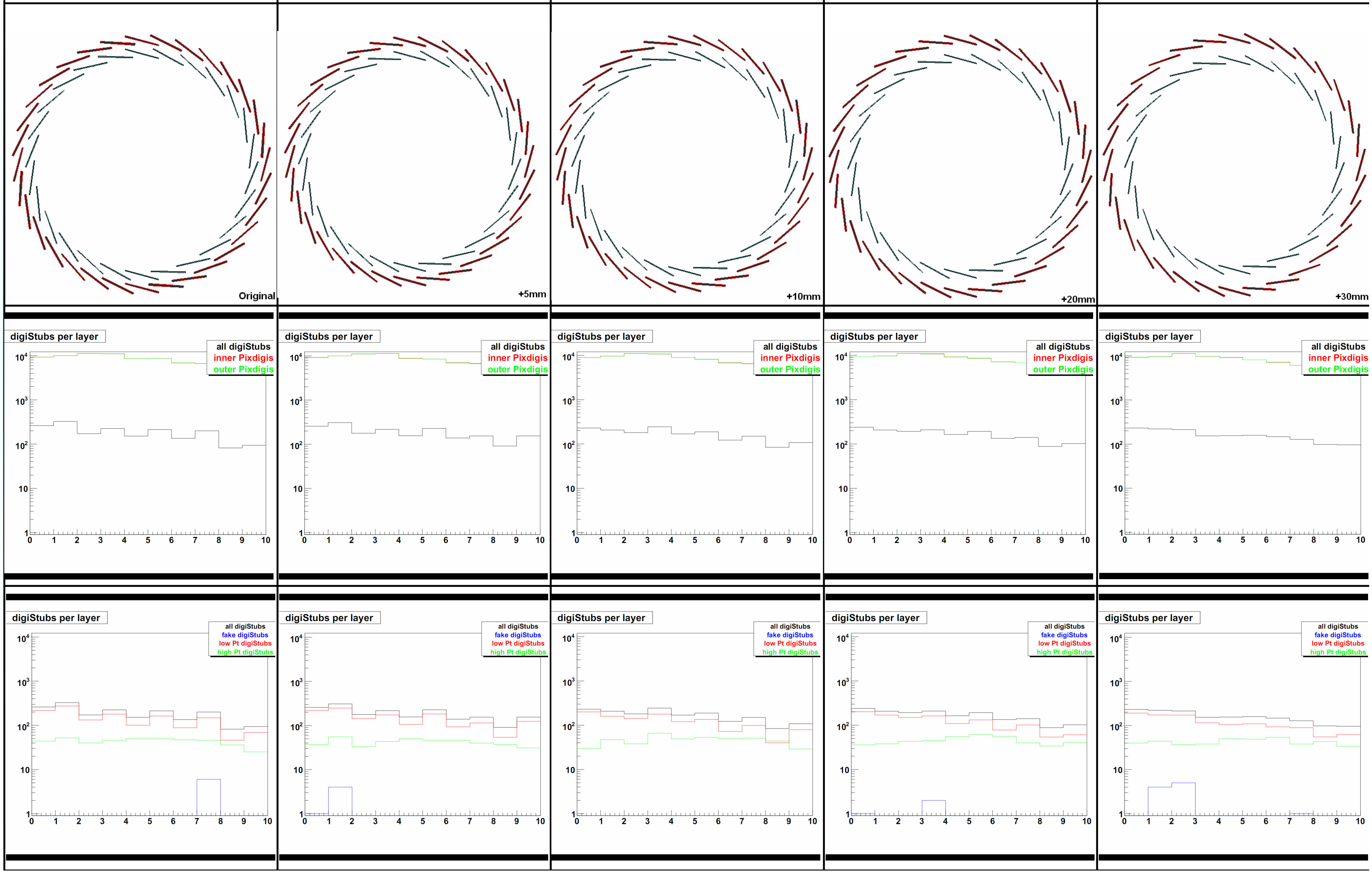


Test: Number of hits should remain ~constant with increasing separation but the number of stubs should fall…

†Please note! This is the inter-stack spacing, not the spacing of elements within a stack!

Some results (v): How many?

Fermilab geometry	Odd layers +5mm	Odd layers +10mm	Odd layers +20mm	Odd layers +30mm



Some results (vi): How many?

Conclusion: We think we understand what is going on here!

Evidence suggests that material interactions in one layer are producing tracks which result in closely bunched hits in the next layer which in turn cause a large number of "acceptable" pairs which are formed into stubs

We are still working on producing definitive proof that this is the case

Some results (vii): How important is clusterization?

An interesting quantity to know is the ratio of the total number of hits in a module to the number of stubs in that module

It is instructive to know this both for stubs from simhits and for stubs from digis

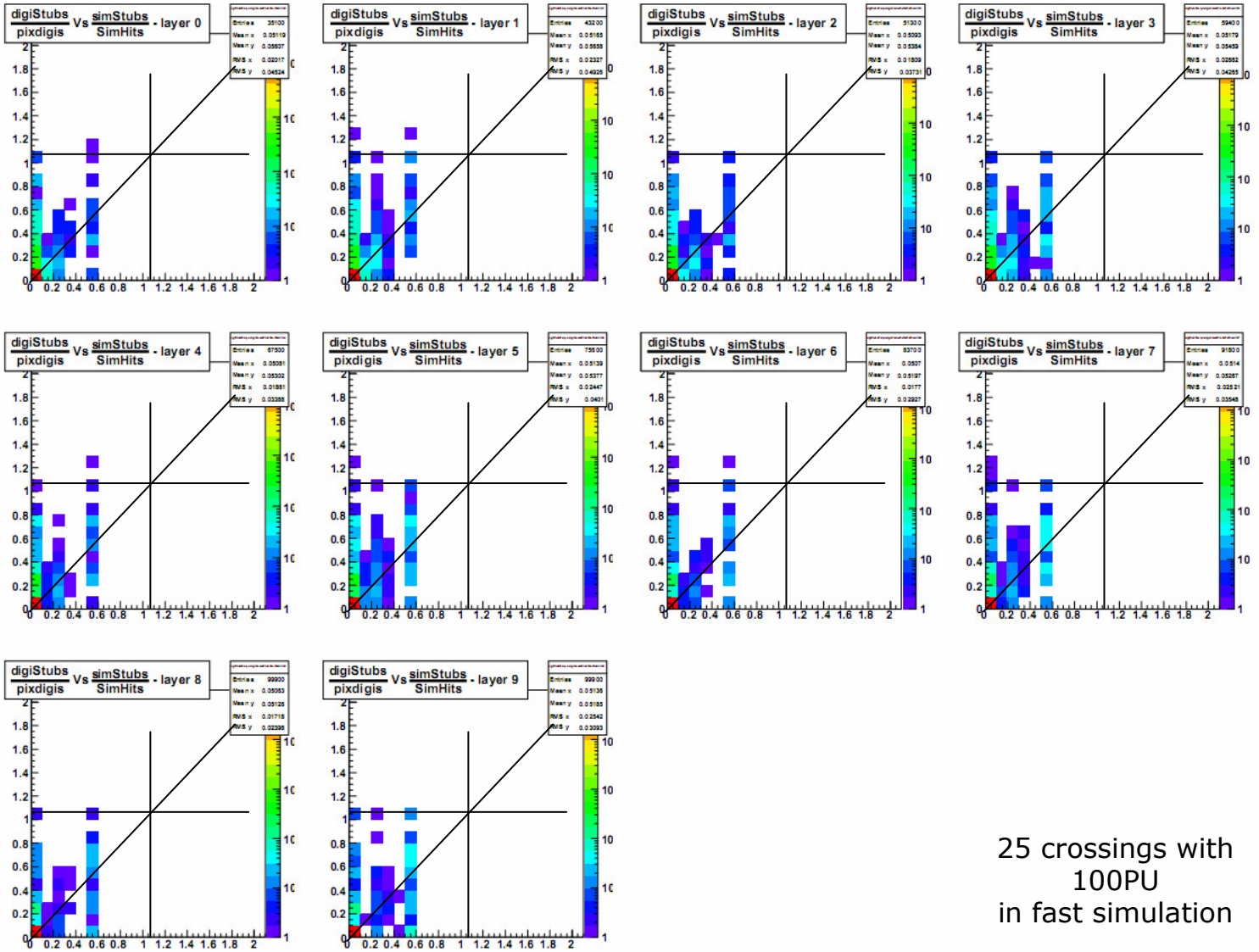
Clearly this will depend on the matching algorithm used, the occupancy and clustering effects (closely correlated tracks, charge sharing, etc)

High pileup should increase the number of hits in line with or faster than the number of stubs whereas large clustering effects should increase the number of stubs faster than the number of hits

Looking for a figure of less than 1 for an overall rate reduction and less than 0.5 from a naïve assumption of pure 2-to-1 mapping of hits to stubs.

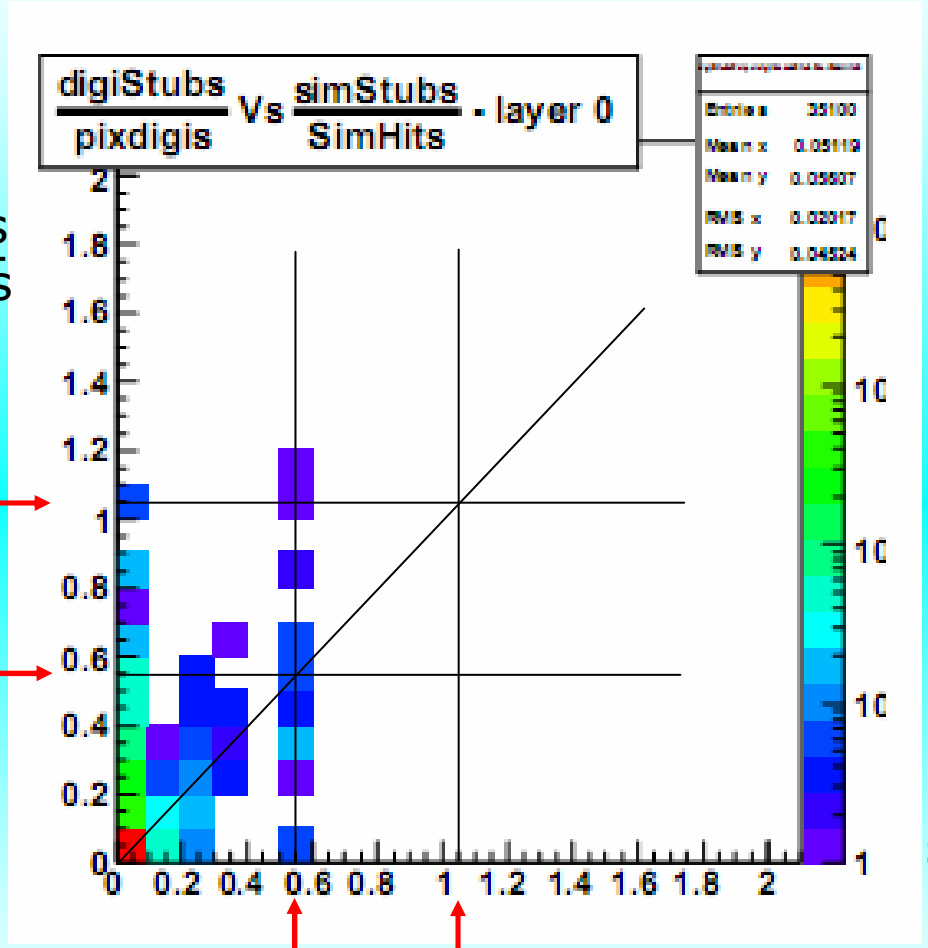
Expect performance of simhits to be better than that of digis

Some results (viii): How important is clusterization?



25 crossings with
100PU
in fast simulation

Some results (ix): How important is clusterization?



digi stubs / pixel digis

Threshold for data reduction

Naïve expectation

25 crossings with 100PU in fast simulation

sim stubs / sim hits

Naïve expectation

Threshold for data reduction

Some results (x): How important is clusterization?

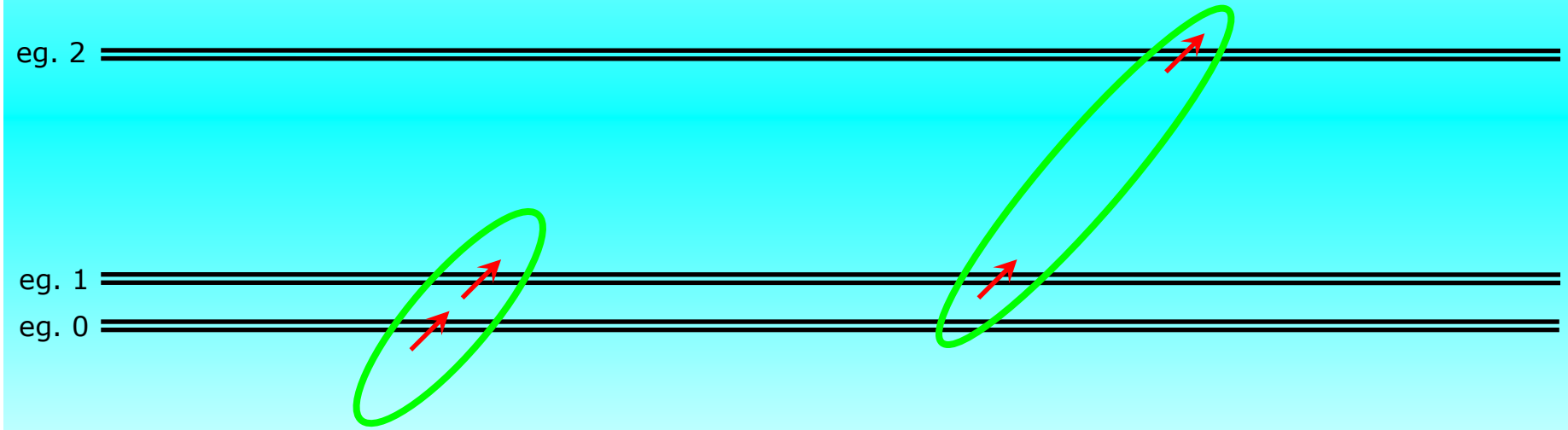
Conclusion: We are just starting to study this.

This study is naïve in certain assumptions and a more detailed analysis is needed to separate the effects of pileup from that of clustering

Some results (xi): Tracklets

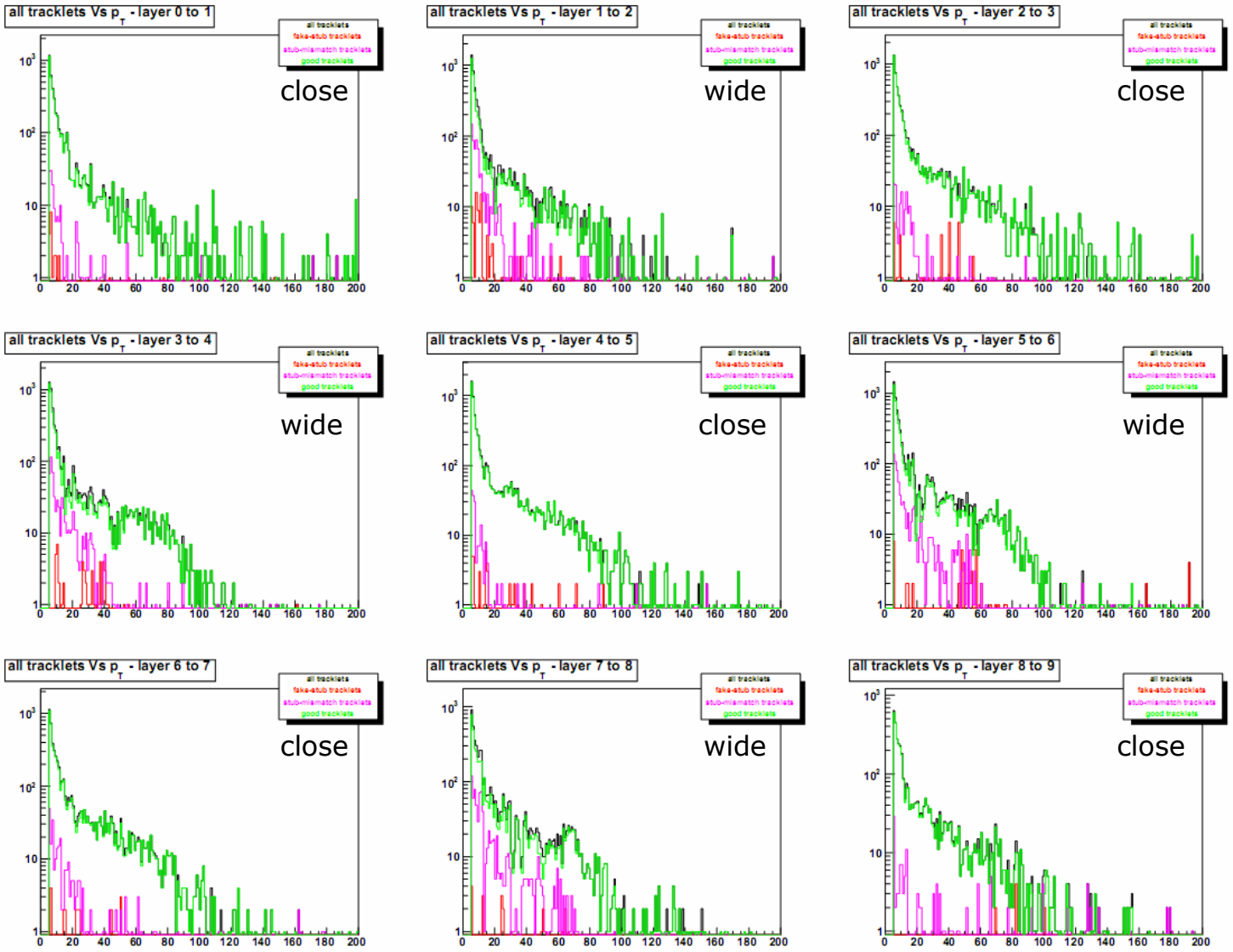
For now we are only considering tracklets formed from stubs in consecutive layers

Recall matching done by placing a cut on ΔΦ based on a p_T threshold and a cut on the projected vertex position



Expect better performance from the closely spaced double-stacks than from the widely spaced double stacks.

Some results (xii): Tracklets



100 crossings with 200PU in fast simulation

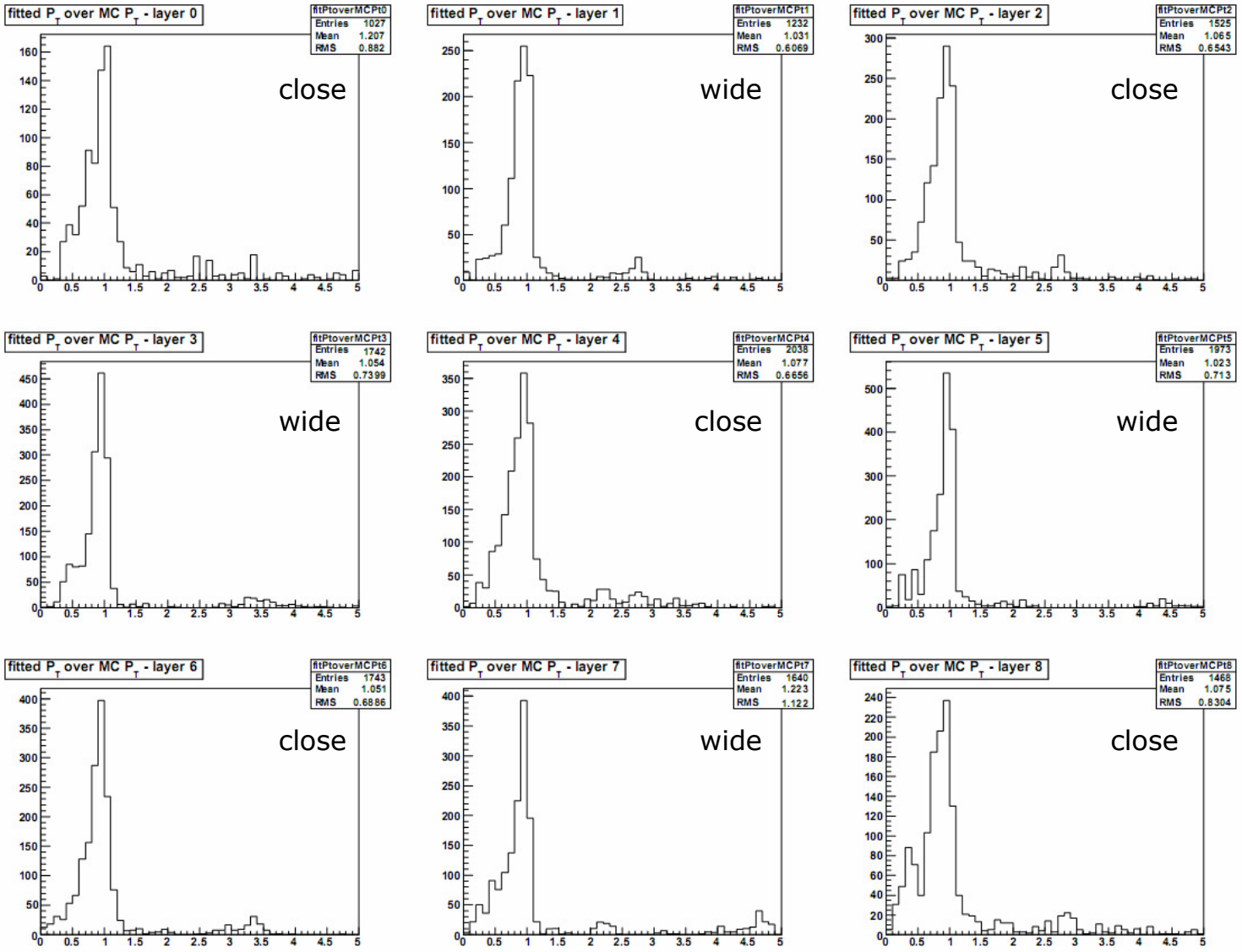
Those where the two stubs are real and the track Ids match

Those where one of the stubs in the tracklet is fake

Those where the two stubs are real but the track Ids do not match

pT here is derived from the best fit helix although using a two point method provides very similar results

Some results (xiii): Tracklets



100 crossings with 20PU in fast simulation

Ratio of fitted p_T to MC p_T for "good" tracklets

pT here is derived from the best fit helix although using a two point method provides very similar results

Some results (xiv): Tracklets

Conclusion: This is very preliminary work!

As expected tracklet formation is more accurate for between the more closely spaced layers

For “true” tracklets, pT performance is better for widely spaced layers, again as expected

Investigation into other properties is on-going

Some results (xv): 'electron trigger'

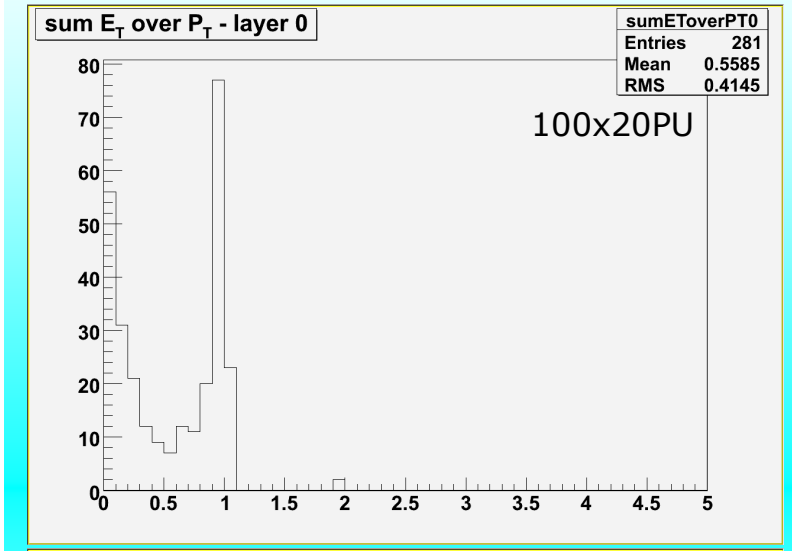
Can stubs be used to seed searches in the calorimeter for a level-1 electron finder?

How much better do tracklets perform?

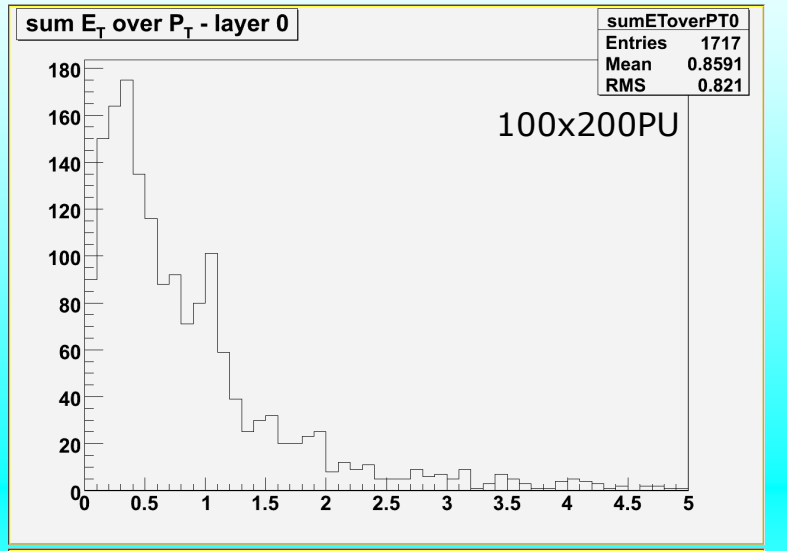
Idea: Project a stub from tracker to ECAL face and compare the reconstructed energy of that ECAL trigger tower to the MC p_T for stubs or fitted p_T for tracklets

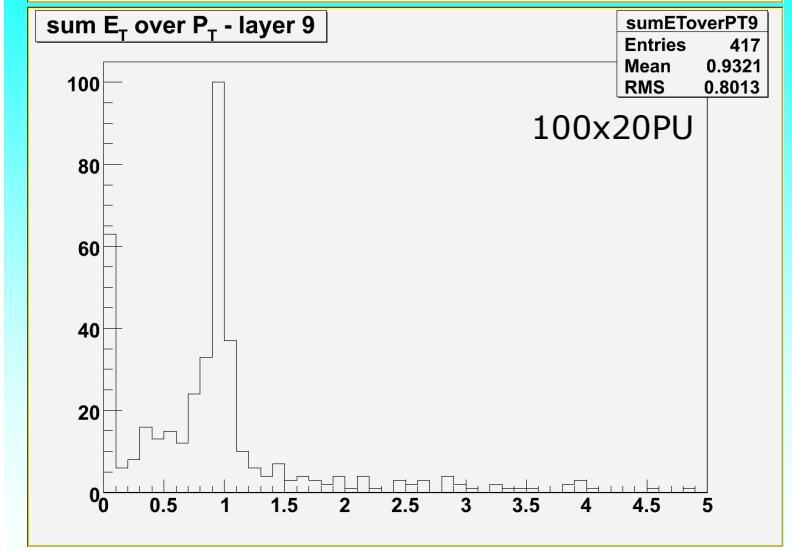
Also look at sum of the projected tower energy with that of the highest energy neighbour like the current L1 trigger

Some results (xvi): 'electron trigger'

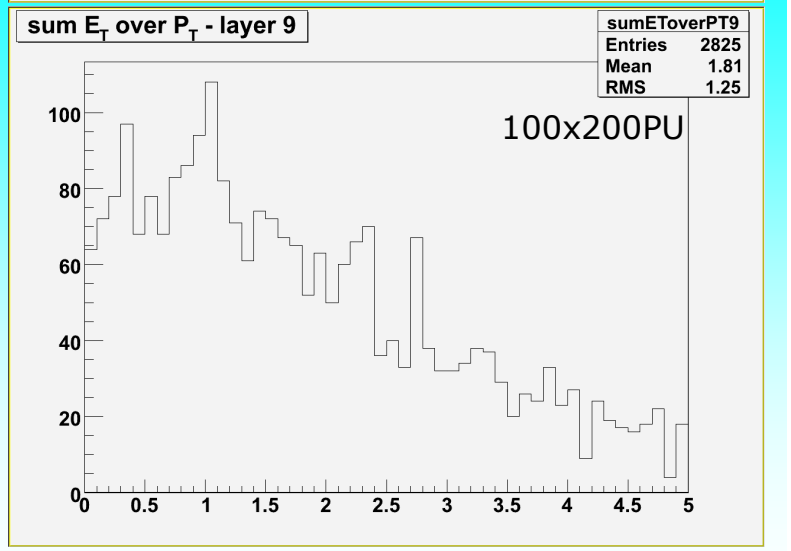


Straight line projection from the nominal vertex, through a digi-type stub to the ECAL face

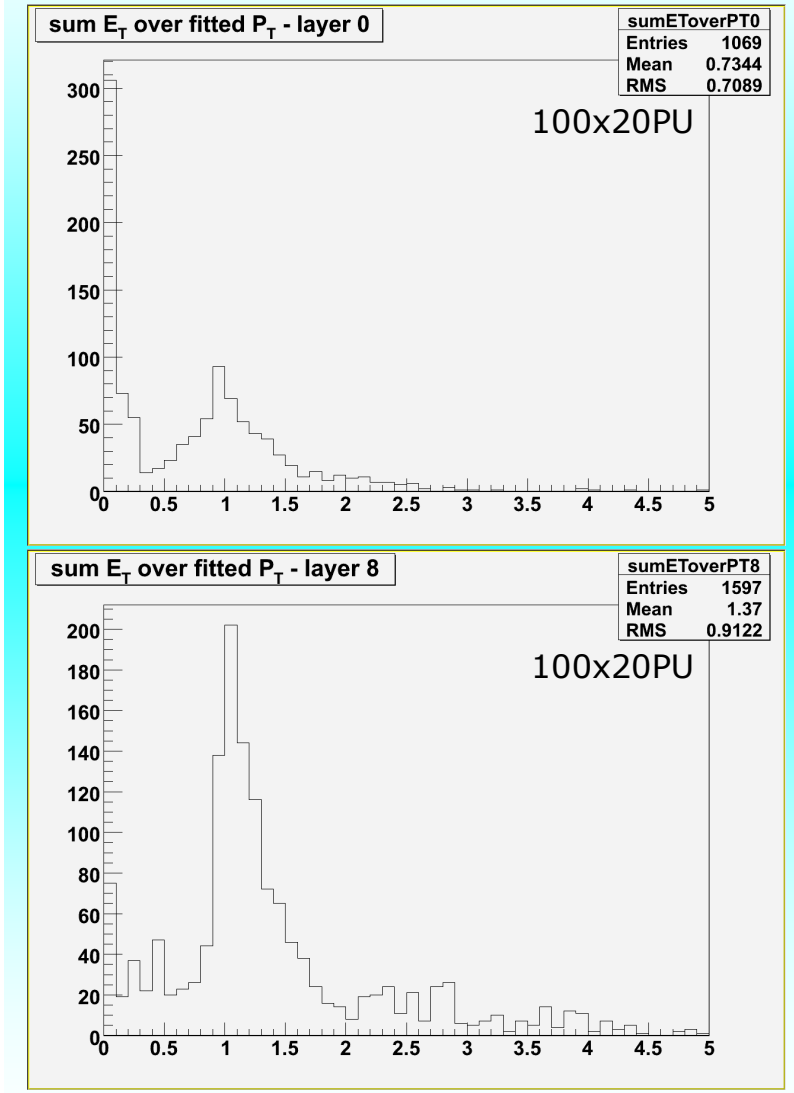




p_T is the MC truth p_T



Some results (xvii): 'electron trigger'



This is very, very preliminary work!
Do not take this as gospel!

Helical projection (derived from the tracklet) from the vertex to the ECAL face

p_T is the fitted tracklet p_T

No 200PU comparison here as simulation failed – trying to understand why

Comparison should not be drawn between this and the previous slide, this is just to show that work has started!

Some results (xviii): 'electron trigger'

Conclusion: This is very preliminary work!

At low (LHC) luminosity a single stub looks sufficient for simple correlations with the ECAL, at high luminosity pileup hinders a simple visual comparison!

Work with tracklets is still at a very preliminary stage!

Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

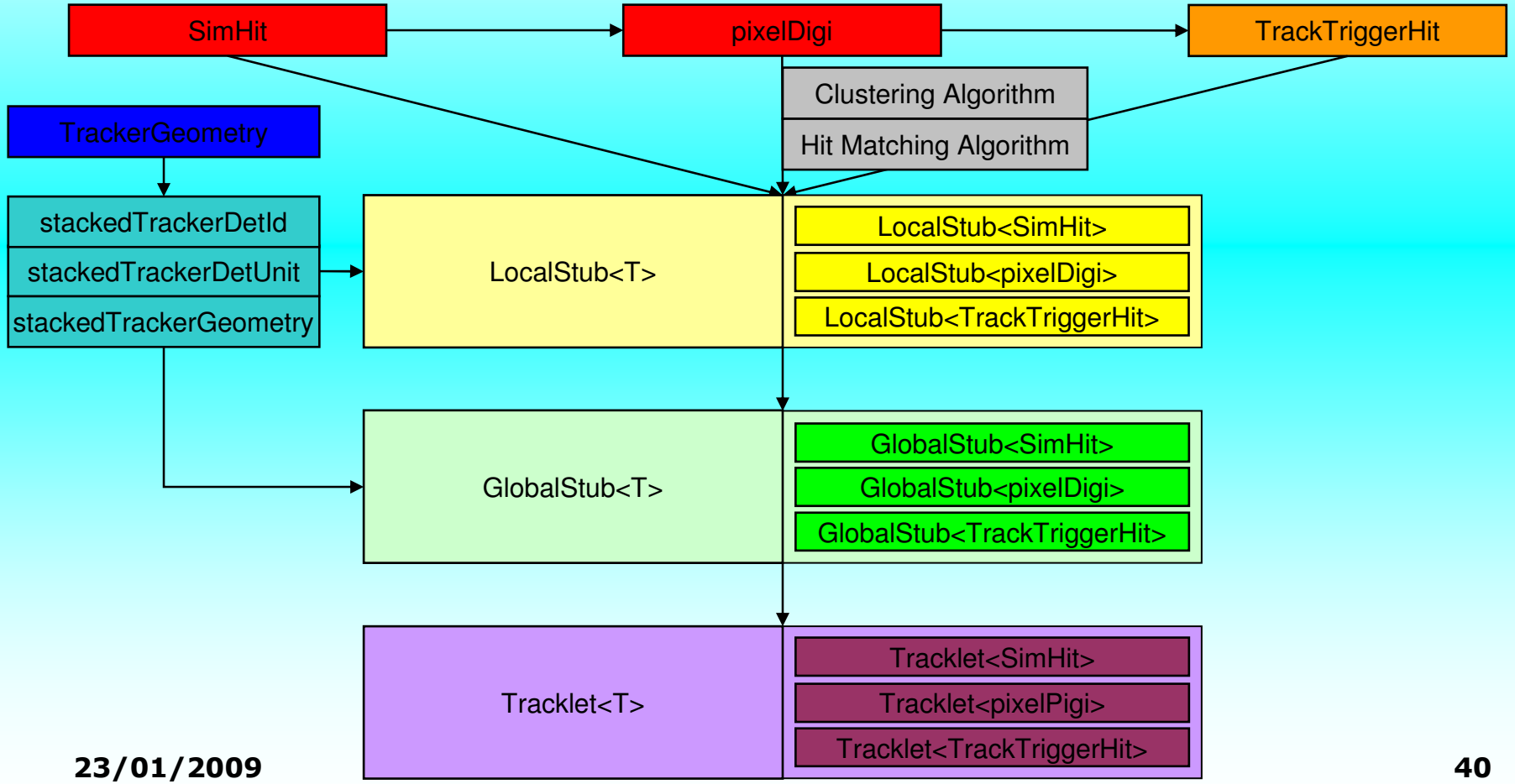
What next…

- Everything currently tested and working in CMSSW_1_8_4 however this is soon to be depreciated!

- Porting to CMSSW_2_2_3 underway
  - Geometry has already been ported
  - Porting of framework digitization code is done but still being tested
  - We have received the preliminary 2_2_3 geometry release and initial tests performed (more complete validation still to do)

What next…

- For CMSSW_2_2_3 we are looking at templating the stub and tracklet classes to reduce the size of the code base and increase code reuse rather than code duplication.



What next…

- Further validation and understanding of the code and geometry
- First large scale data production in CMSSW_2_2_3
- Analysis!

Summary

- A little terminology
- Simulation infrastructure
- Some results
- What next…
- Conclusion

Conclusion

- The tools have been tested in CMSSW_1_8_4 for both fast and full sim and the data formats have been written to event file and read back
- The tools are being ported to CMSSW_2_2_3 in preparation for the first large scale data production
- Code exists for the analysis of framework objects (ie geometry, hits, stubs, tracklets, etc)
- Studies of performance relevant to L1 triggers have commenced

The geometry utilities are described at
https://twiki.cern.ch/twiki/bin/view/CMS/SLHCStackedTrackerTools

Instructions on how to use the geometry utilities can be found at
https://twiki.cern.ch/twiki/bin/view/CMS/SLHCStackedTrackerToolsTutorial

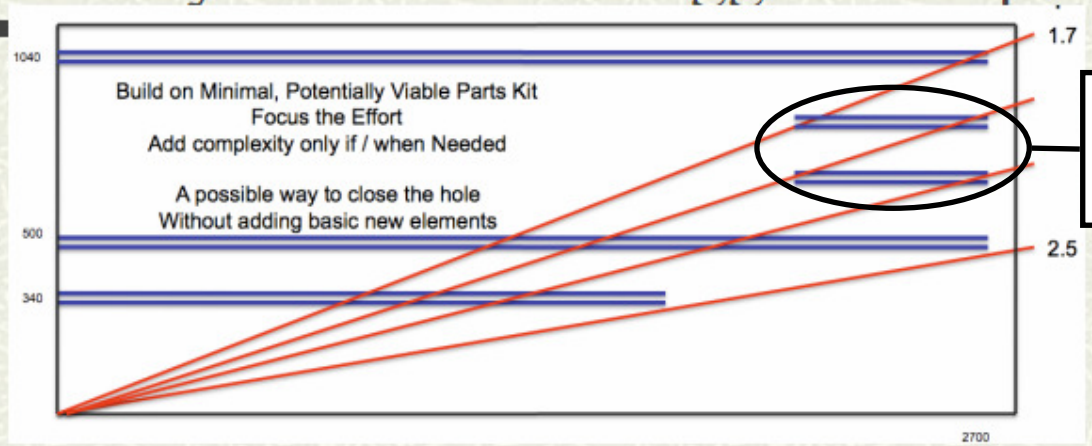
More info about data formats and instructions on how to generate stubs can be found at
https://twiki.cern.ch/twiki/bin/view/CMS/TrackTriggerHitsAndStubs

Spares

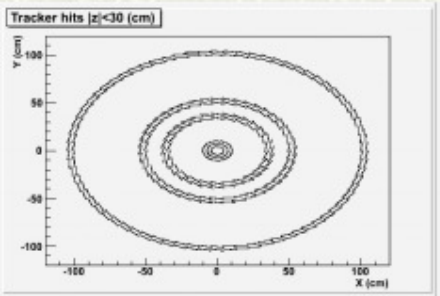
Current geometry model status

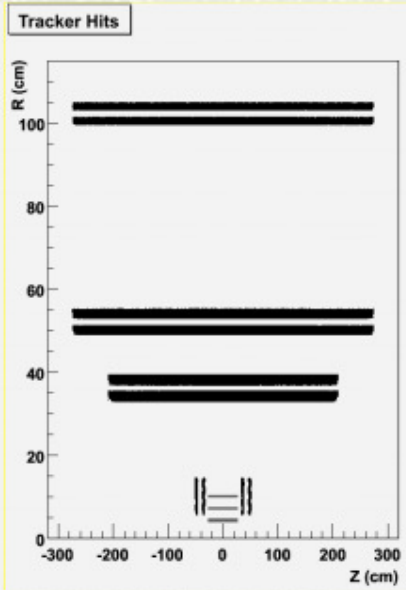
Baseline Geometry from Track Trigger Group

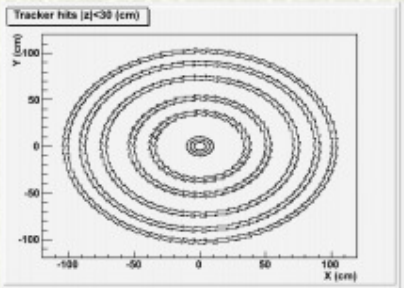
Start with StrawB
Remove TEC
Add in extra Layers
Extend Barrel Length

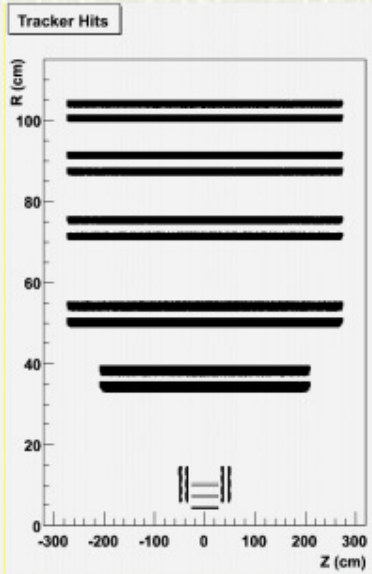


Short layers









Mike Weinberger 8