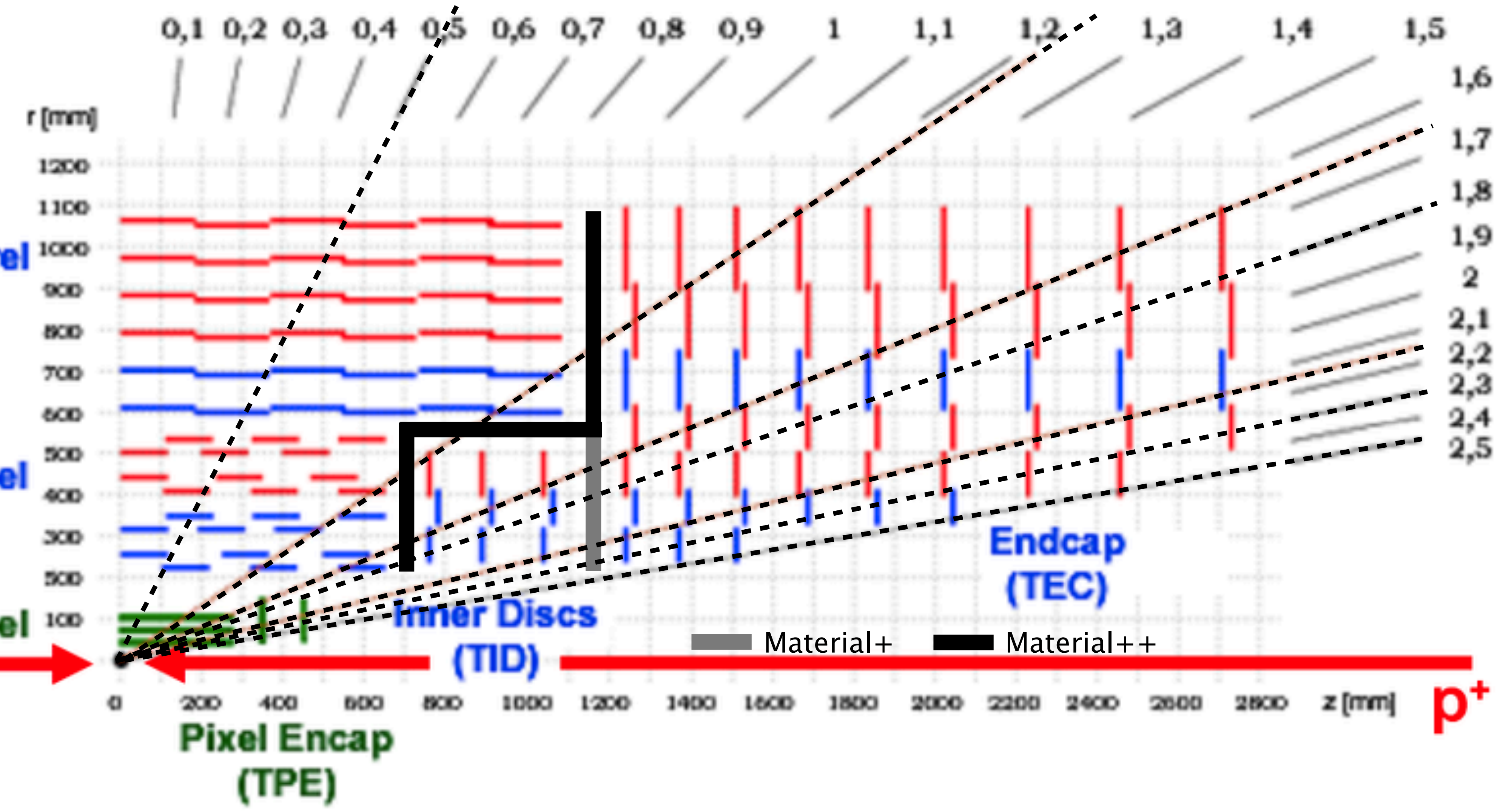
Track Seeding, Track Selection, Iterative Tracking (and all that)

Tracking Training Day – Oct. 18, 2013

G.Cerati (UCSD)

Tracker Layout (ZR plane)



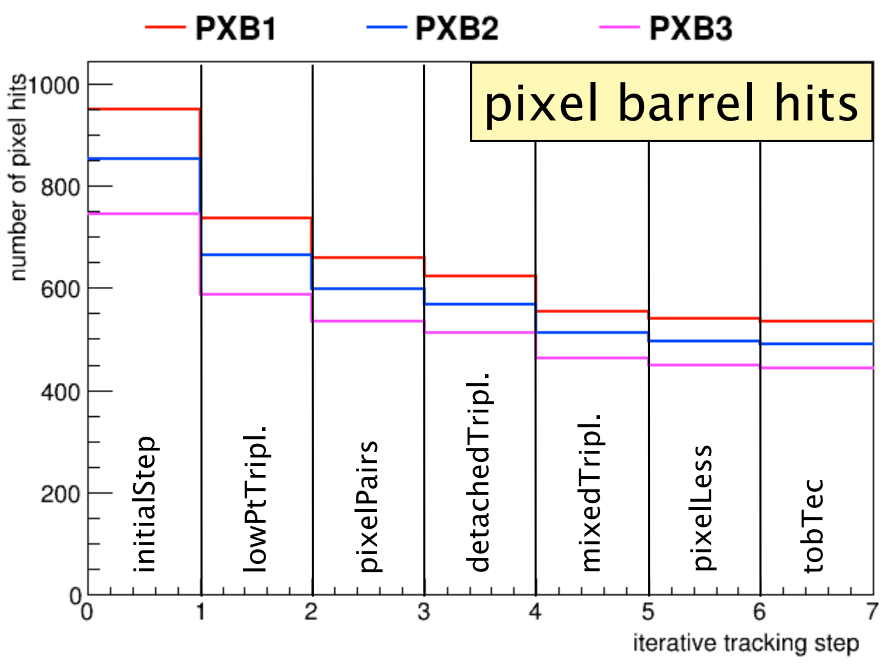
for reference

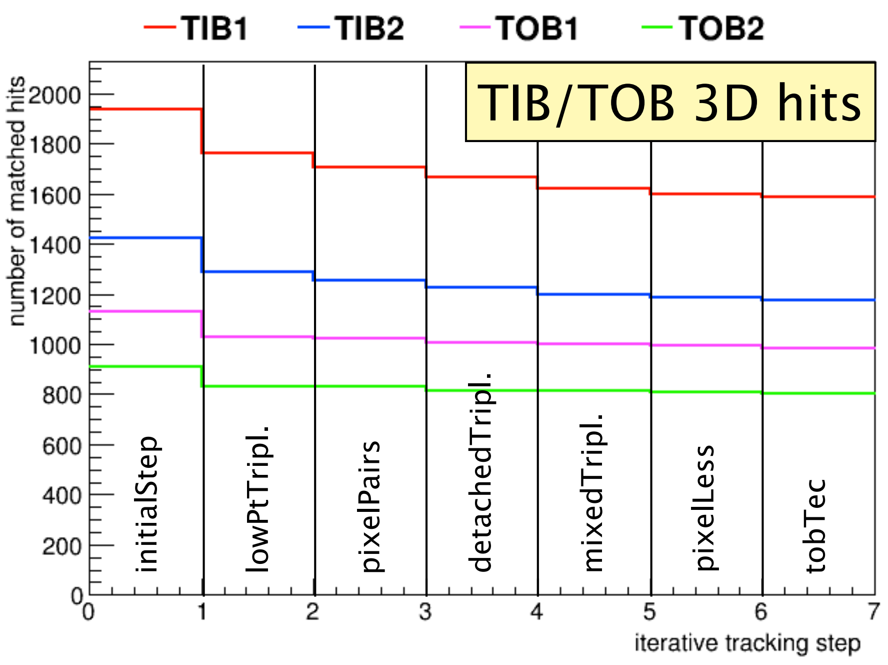
Introduction to Iterative Tracking

Iterative Tracking

- Iterative tracking philosophy: Run track reconstruction several times, removing used hits from consideration, in order to reconstruct tracks which would normally be prohibitively expensive in CPU to find.
- In general, seeding and track building using triplets of pixel hits which must originate from a vertex or beamspot and have high momentum (>0.7 GeV) is computationally easy.
  - Removing these constraints increases the combinatorics.
- The combinatorics can be reduced by not including already found hits but there are lots of hits that are NEVER found (lots of low p_T tracks) so even at the end, there are lots of combinations.
- Loosening up the vertex/beamspot constraint adds the most to the combinatorics but is necessary for two reasons:
  - To find displaced tracks from conversions, nuclear interactions, and beauty/charm/strange decays
  - To find prompt tracks that are not found by pixel seeding but ARE found in strip seeding. Extrapolation from pairs of strips to beamspot is not precise so need wider search window.

Size of the combinatorial problem

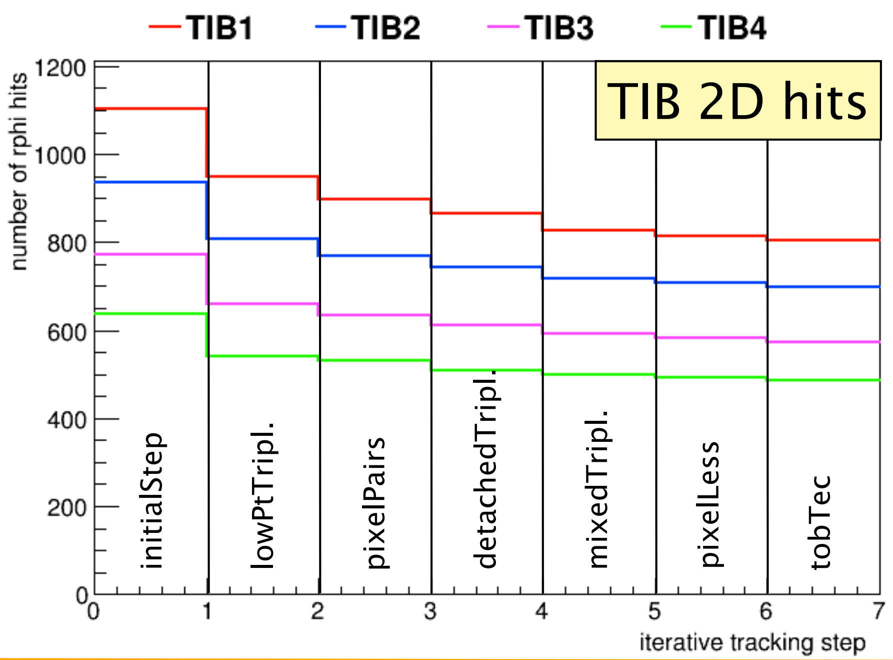




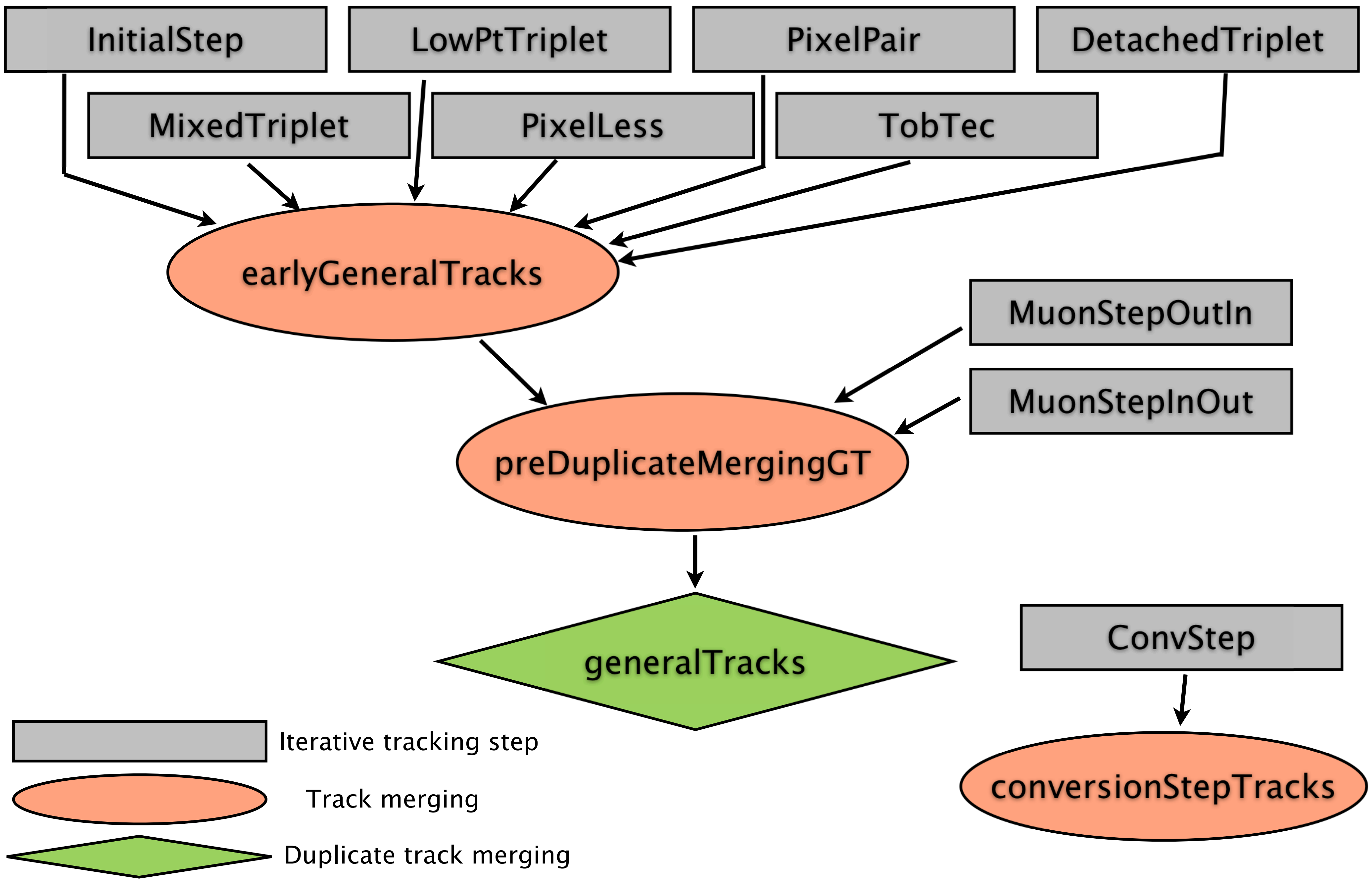
Sample: TTbar+20 PU

First iteration is most effective in reducing the number of hits. After all iterations, more than half of the hits are still not associated to tracks.

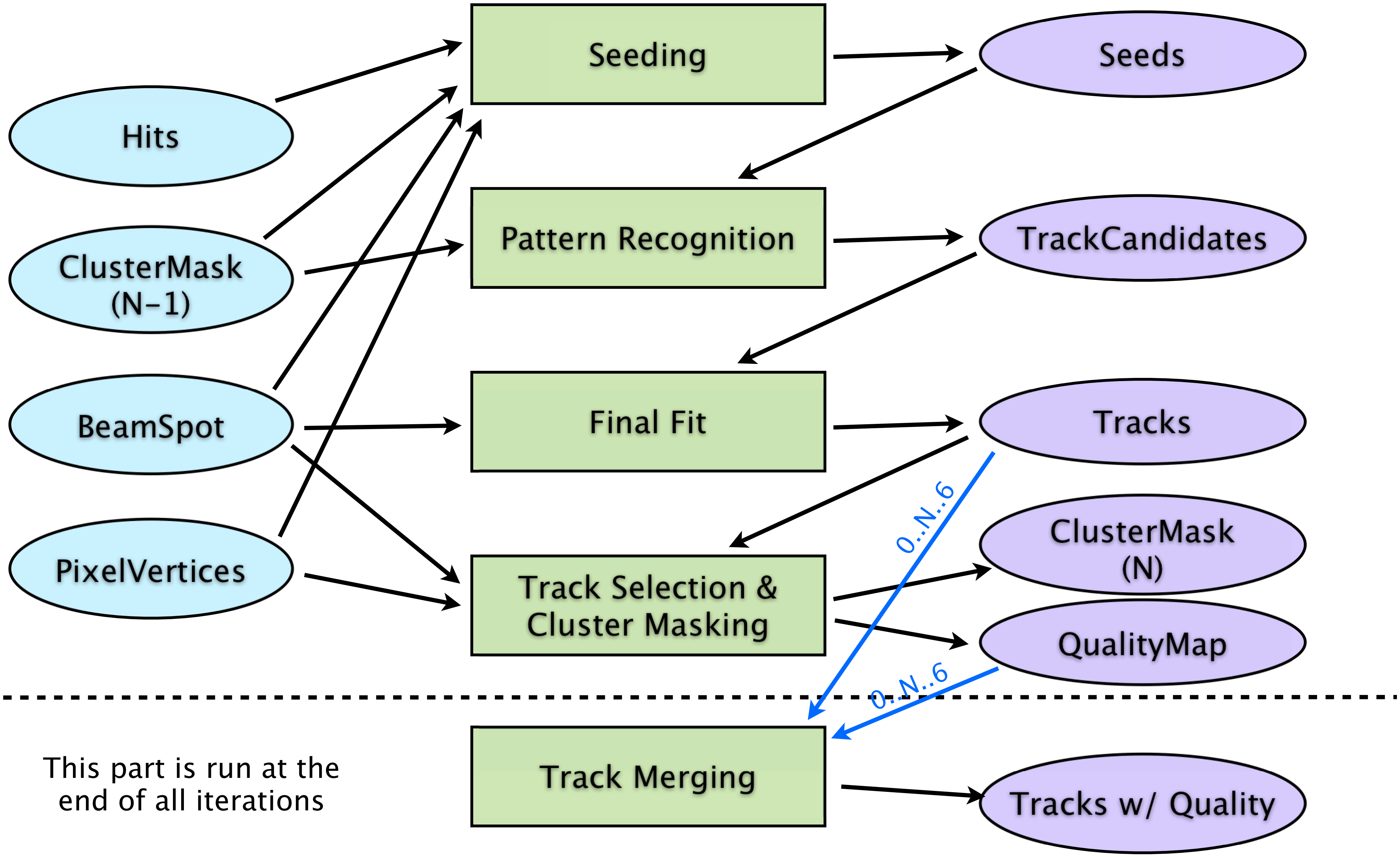
Iterative tracking approach reduces the combinatorics but it's still a challenge.



Iterative Tracking Sequence



Modules for Nth iteration



This part is run at the end of all iterations

Summary of Iterations (seeding, CMSSW700pre4)

Iteration	Target	Seeding	p_T>	d_0<	d_z<
iter0 InitialStep	high p_T prompt	pixel triplets	0.6 GeV	0.020 cm	$4.0\sigma_{BS}$
iter1 LowPtTriplets	low p_T prompt	pixel triplets	0.2 GeV	0.020 cm	$4.0\sigma_{BS}$
iter2 PixelPairs	recover efficiency for prompt	pixel pairs + vertex	0.6 GeV	0.015 cm	$3.0\sigma_{vtx}$
iter3 DetachedTripl.	displaced-- tracks	pixel triplets	0.3 GeV	1.5 cm	15 cm
iter4 MixedTriplets	displaced- tracks	pixel-strip mixed triplets	0.6 (0.4) GeV	1.5 cm	10 (15) cm
iter5 PixelLess	displaced+ tracks	TIB-TID-TEC pairs	0.7 GeV	2.5 cm	15 cm
iter6 TobTec	displaced++ tracks	TOB-TEC pairs	0.6 GeV	6.0 cm	30 cm
iter9 InOut	promote quality to global mus	standalone muons	2 GeV	-	-
iter10 OutIn	recover trk efficiency for mus	tracker muons	10 GeV	-	-

note: barrel (forward)

Iterative Tracking Configuration Files

- How could you have figured that out yourself ?

- Look at a tracking iteration inside RecoTracker/IterativeTracking/python/iterativeTk_cff.py

- E.g. Iteration 2 (PixelPairStep_cff.py) contains:
  - pixelPairStepSeeds = … GlobalSeedsFromPairsWithVertices_cff …
  - pixelPairStepSeeds.RegionFactoryPSet.RegionPSet.ptMin = 0.6
  - pixelPairStepSeeds.RegionFactoryPSet.RegionPSet.originRadius = 0.015
  - pixelPairStepSeeds...SeedingLayers = cms.string('pixelPairStepSeedLayers')

- Where seeding layers are:
  - pixelPairStepSeedLayers = cms.ESProducer("SeedingLayersESProducer",
  - ComponentName = cms.string('pixelPairStepSeedLayers'),
  - layerList = cms.vstring('BPix1+BPix2', 'BPix1+BPix3', 'BPix2+BPix3' …)

Iterative Tracking Configuration Files

- How could you have figured that out yourself ?
- Look at a tracking iteration inside RecoTracker/IterativeTracking/python/iterativeTk_cff.py
- E.g. Iteration 2 (PixelPairStep_cff.py) contains:
  - pixelPairStepSeeds = … GlobalSeedsFromPairsWithVertices_cff …
  - pixelPairStepSeeds.RegionFactoryPSet.RegionPSet.ptMin = 0.6
  - pixelPairStepSeeds.RegionFactoryPSet.RegionPSet.originRadius = 0.015
  - pixelPairStepSeeds…SeedingLayers = cms.string('pixelPairStepSeedLayers')
- Where seeding layers are:
  - pixelPairStepSeedLayers = cms.ESProducer("SeedingLayersESProducer",
  - ComponentName = cms.string('pixelPairStepSeedLayers'),
  - layerList = cms.vstring('BPix1+BPix2', 'BPix1+BPix3', 'BPix2+BPix3' …)

For the hands-on session!

Summary of Iterations (building, CMSSW700pre4)

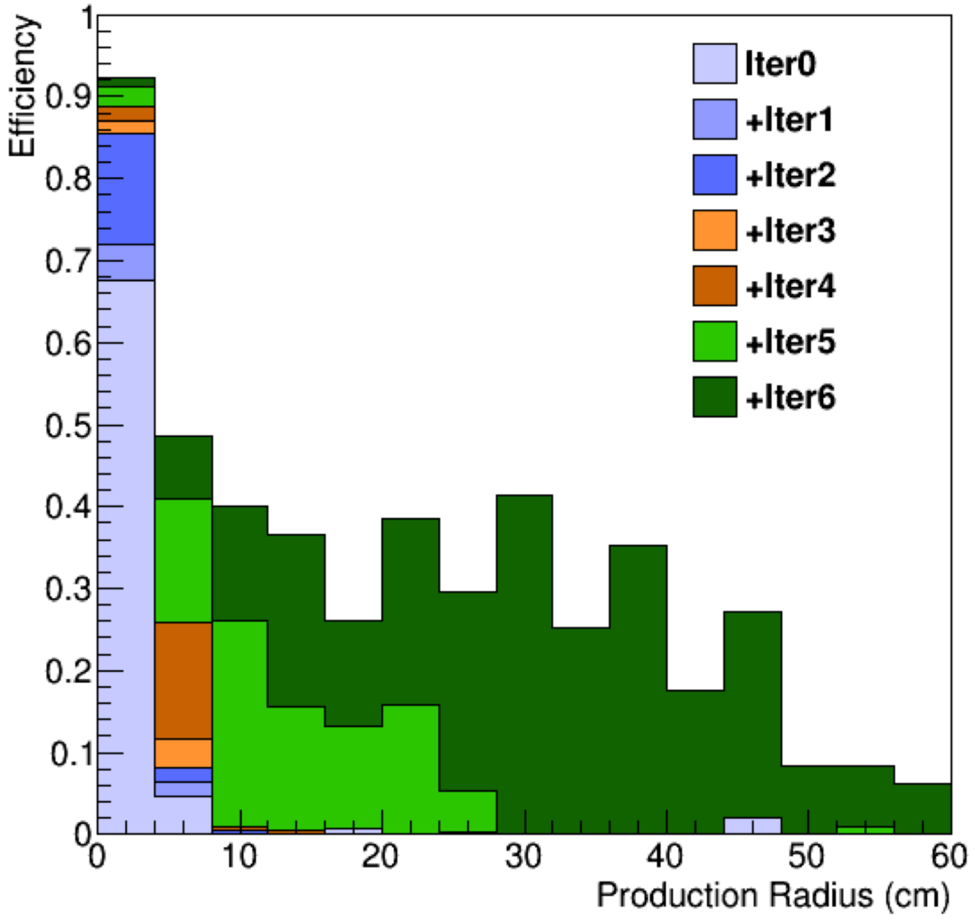
Iteration	Target	Nhits≥	Nlost≤	hit χ^2	Ncand/seed
iter0 InitialStep	high p_T prompt	3	1+0.1·Nhits	30	6
iter1 LowPtTriplets	low p_T prompt	3	1+0.1·Nhits	9	4
iter2 PixelPairs	recover efficiency for prompt	3	1+0.1·Nhits	9	3
iter3 DetachedTripl.	displaced-- tracks	3	0.7+0.1·Nhits	9	2
iter4 MixedTriplets	displaced- tracks	3	0	16	2
iter5 PixelLess	displaced+ tracks	4	0	9	2
iter6 TobTec	displaced++ tracks	4 inOut 6 outIn	0	16	2
iter9 InOut	promote quality to global mus	3	10+0.1·Nhits	400	5
iter10 OutIn	recover trk efficiency for mus	5	10+0.1·Nhits	30	3

Summary of Iterations (highPurity, CMSSW700pre4)

Iteration	Target	$N_{Layers} \geq$	$N_{3DLay} \geq$	$N_{LostLay} \leq$	$\chi^2/ndof \leq$	PV compat.
iter0 InitialStep	high p_T prompt	3	3	2	$0.7 \cdot N_{Lay}$	tight
iter1 LowPtTriplets	low p_T prompt	3	3	2	$0.7 \cdot N_{Lay}$	tight
iter2 PixelPairs	recover efficiency for prompt	3	3	2	$0.7 \cdot N_{Lay}$	tight
iter3 DetachedTripl.	displaced-- tracks	3 (5)	3 (4)	1 (0)	$0.7\ (0.3) \cdot N_{Lay}$	medium (loose)
iter4 MixedTriplets	displaced- tracks	3 (5)	3 (4)	1 (0)	$0.4\ (0.25) \cdot N_{Lay}$	medium (loose)
iter5 PixelLess	displaced+ tracks	4	3	0	$0.3 \cdot N_{Lay}$	loose
iter6 TobTec	displaced++ tracks	5	2	0	$0.2 \cdot N_{Lay}$	loose
iter9 InOut	promote quality to global mus	5	2	2	$0.4 \cdot N_{Lay}$	-
iter10 OutIn	recover trk efficiency for mus	5	2	2	$0.4 \cdot N_{Lay}$	-

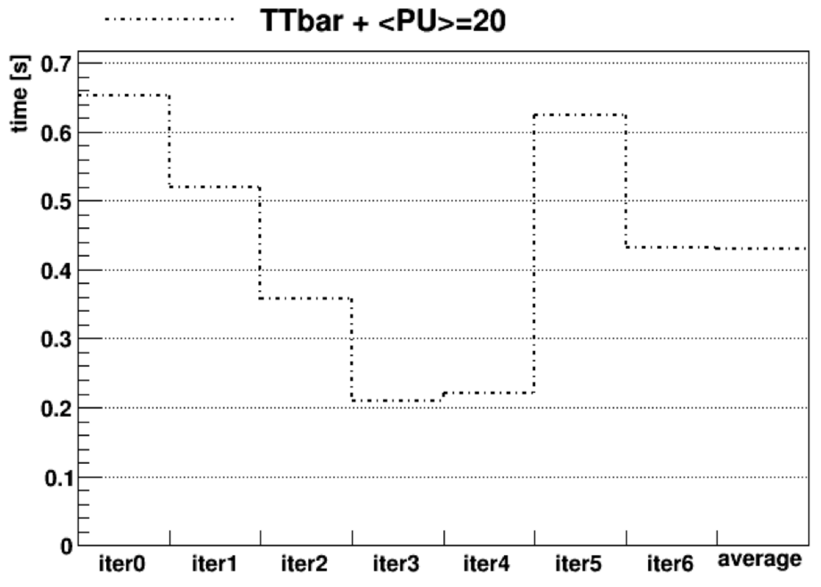
Performance of the iterative steps

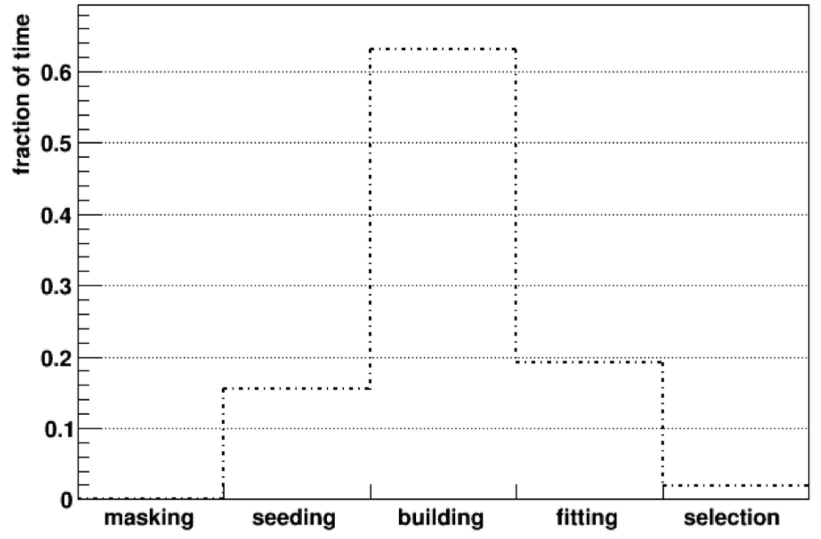
Efficiency vs production radius



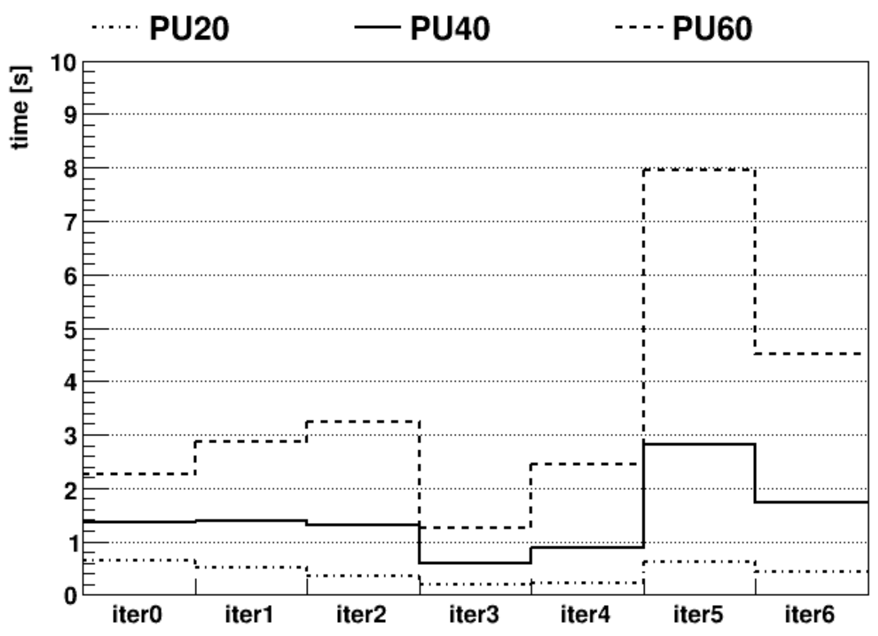
CMSSW_6_2_0, 2012 tracking

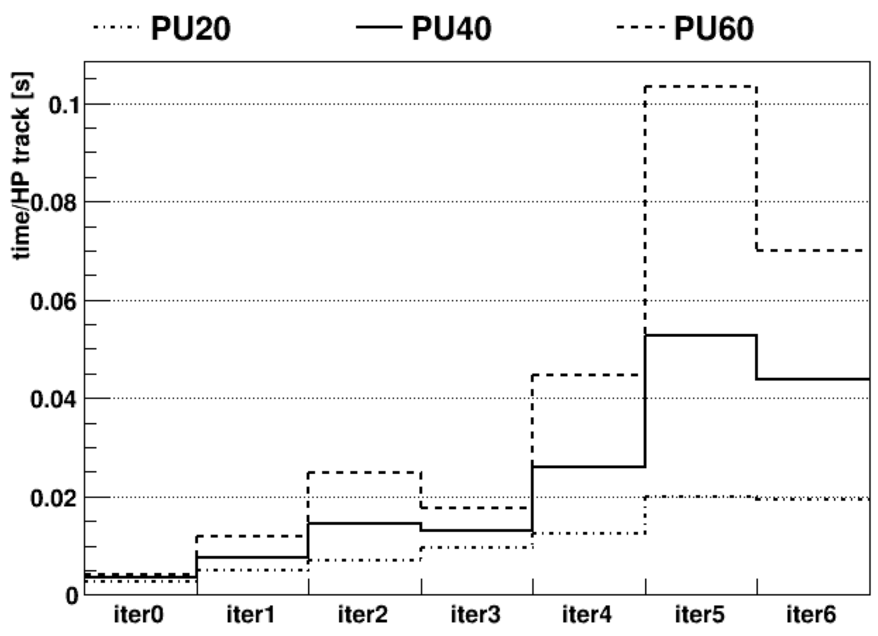
CPU time per iteration and per task





Projections at different PU





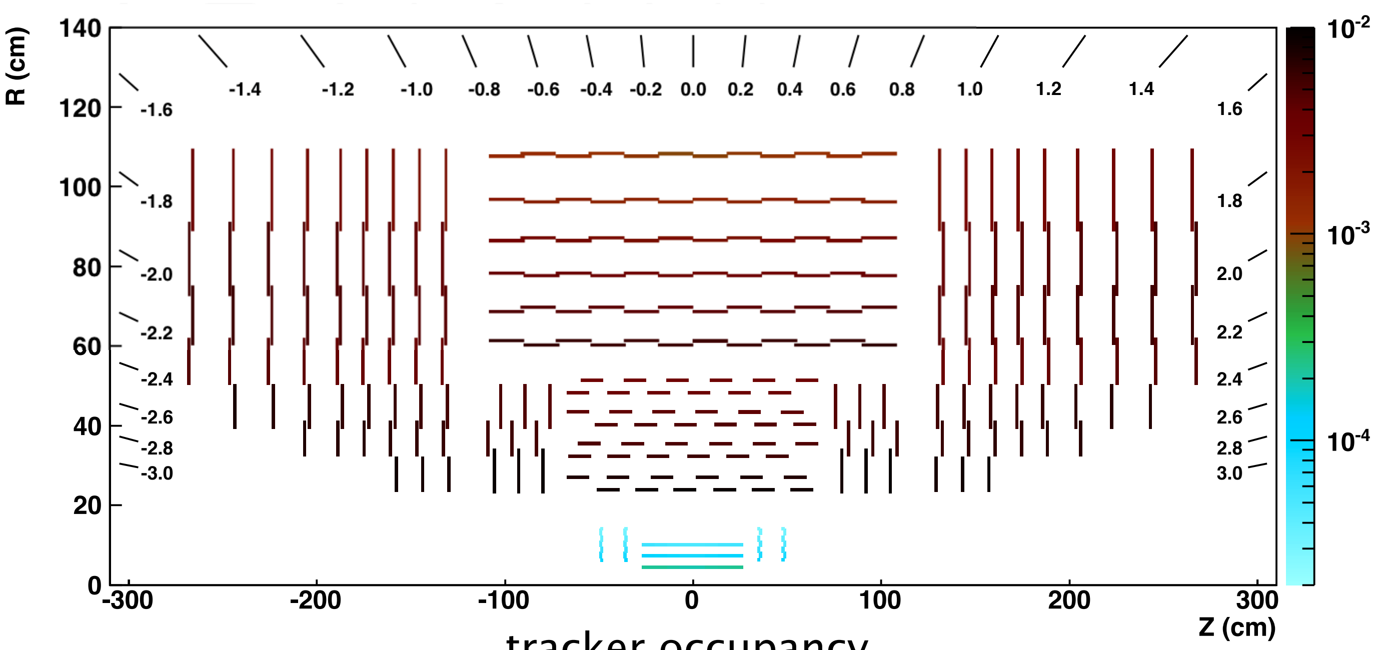
Left plot is average time per event, right plot is time per HighPurity track.
2012 tracking on ttbar events, sqrt(s)=7 TeV, 50 ns bunch crossing.

Only iteration scaling linearly with PU is iter0.
Still reasonable scaling: iter1, iter3.
Dangerous scaling: iter2, iter4, iter5 and iter6.

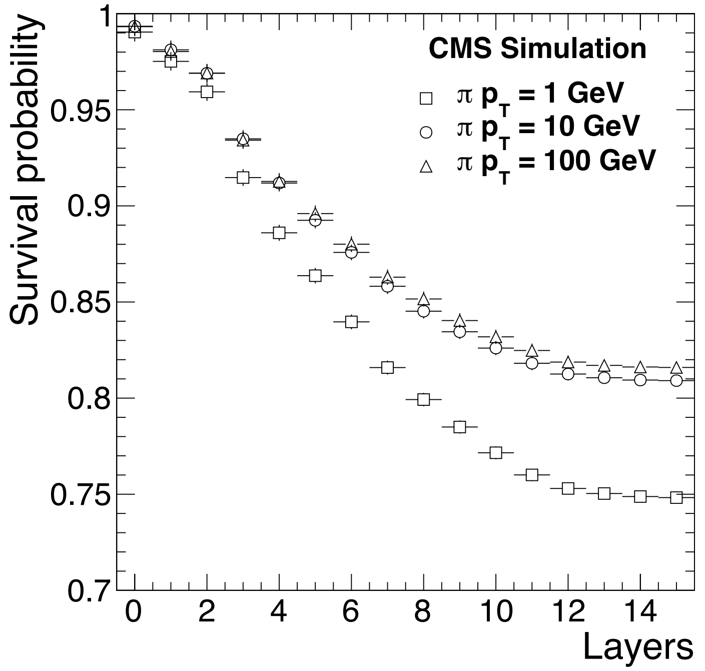
Seeding

Seeding Strategy

- Two features drive the seeding strategy:
  - Tracker Material: a significant fraction of prompt tracks do not reach outer layers due to nuclear interactions
  - Occupancy: lower occupancy layers will provide a cleaner seed collection due to a lower combinatorial background
- The pixel tracker provides high resolution 3D hits close to the interaction point and has the lowest occupancy:
  - ideal for seeding and it is used for most iterations.
- Matched hits in the strip tracker stereo layers complement pixel seeding for the reconstruction of displaced tracks
  - not optimal: poor z-resolution (1 mm) and suffer from “ghost” hits.

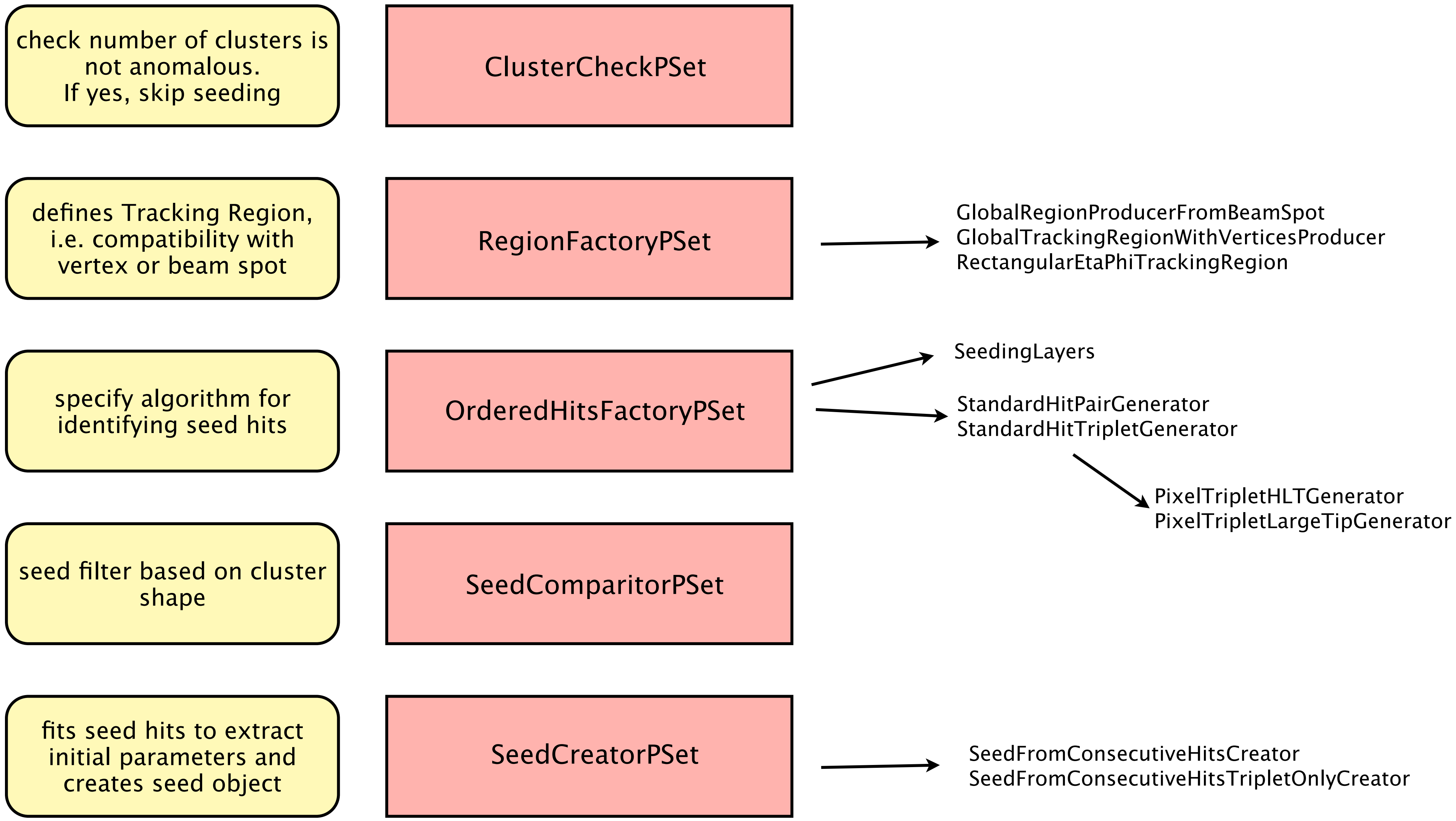


tracker occupancy



Seeding code structure

The **SeedGeneratorFromRegionHitsEDProducer** module can be configured with the following options:


check number of clusters is not anomalous. If yes, skip seeding
ClusterCheckPSet
defines Tracking Region, i.e. compatibility with vertex or beam spot
RegionFactoryPSet
GlobalRegionProducerFromBeamSpot
GlobalTrackingRegionWithVerticesProducer
RectangularEtaPhiTrackingRegion
specify algorithm for identifying seed hits
OrderedHitsFactoryPSet
SeedingLayers
StandardHitPairGenerator
StandardHitTripletGenerator
PixelTripletHLTGenerator
PixelTripletLargeTipGenerator
seed filter based on cluster shape
SeedComparitorPSet
fits seed hits to extract initial parameters and creates seed object
SeedCreatorPSet
SeedFromConsecutiveHitsCreator
SeedFromConsecutiveHitsTripletOnlyCreator

Tracking Regions

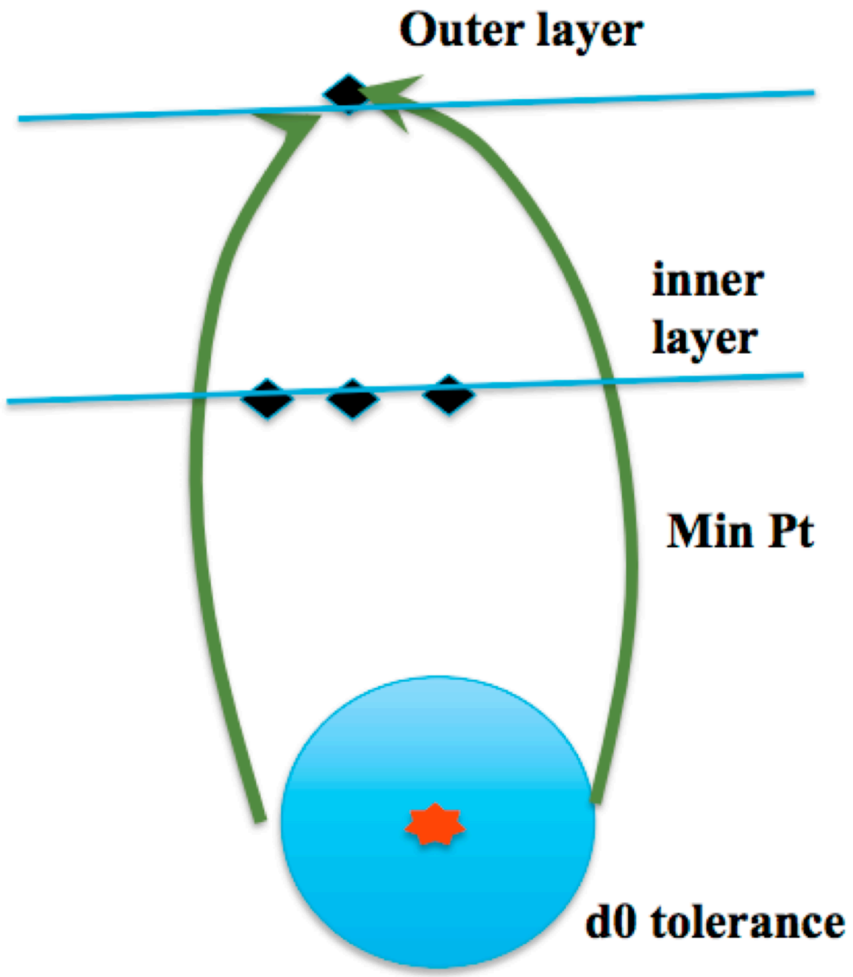
- A TrackingRegion is a phase space region where we intend to reconstruct tracks, where phase space may include both track parameters and the detector global coordinates

- GlobalTrackingRegion:
  - used for offline tracking, works on all directions from an origin point (with tolerance) and a transverse momentum range
  - origin point can be beam spot or pixelVertices
  - tolerance can be defined in absolute terms or as a function of $\sigma_{Z,BS}$ ($\sigma_{Z,vtx}$)
  - p_T range can be defined as minimum p_T or as range of inverse p_T

- RectangularEtaPhiTrackingRegion:
  - used for tracking around jet/lepton direction at HLT
  - add the following constraint:
    - allowed η tolerance around the direction (at vertex)
    - allowed ϕ tolerance around direction (at vertex)

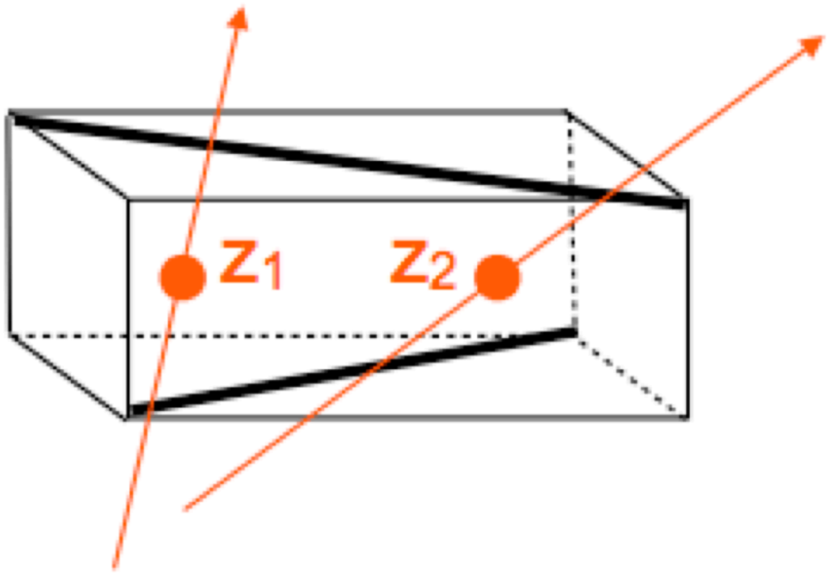
Seed Pairs

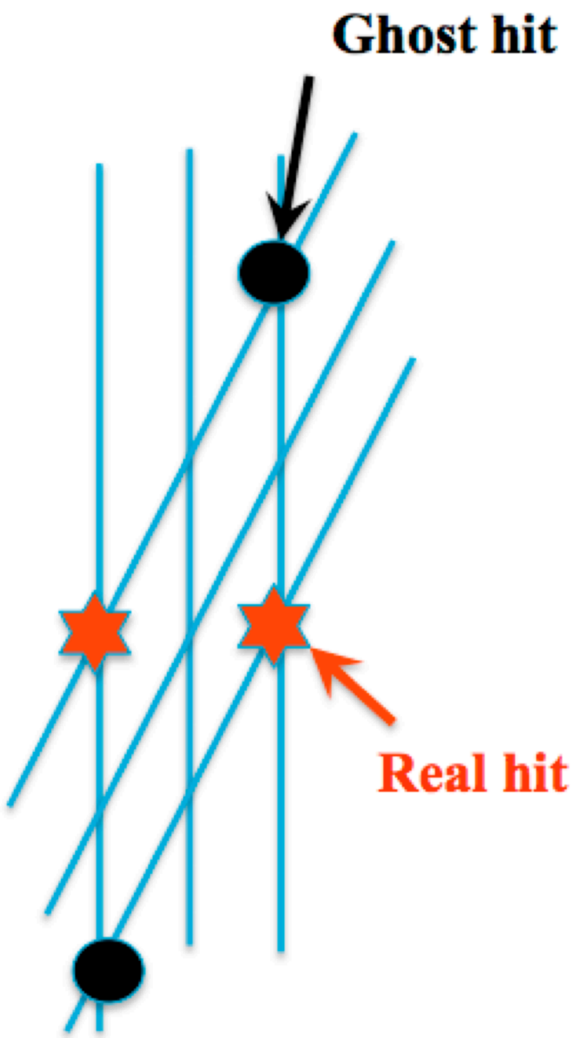
- Loop over hits in outer of two layers.
- For each outer hit, calculate window in inner layer compatible with origin point tolerance and p_T range.
- Build pairs using all hits inside this window.
  - HitPairGeneratorFromLayerPair
- Pairs can be input for triplet search or can define a seed.
- Estimate seed parameters at outer hit from Kalman Filter forward fit of the two hits.
  - fit initial state defined by helix of the two hits + origin point; initial uncertainty defined by d_0, d_z and p_T tolerance.



Strip Seed Pairs

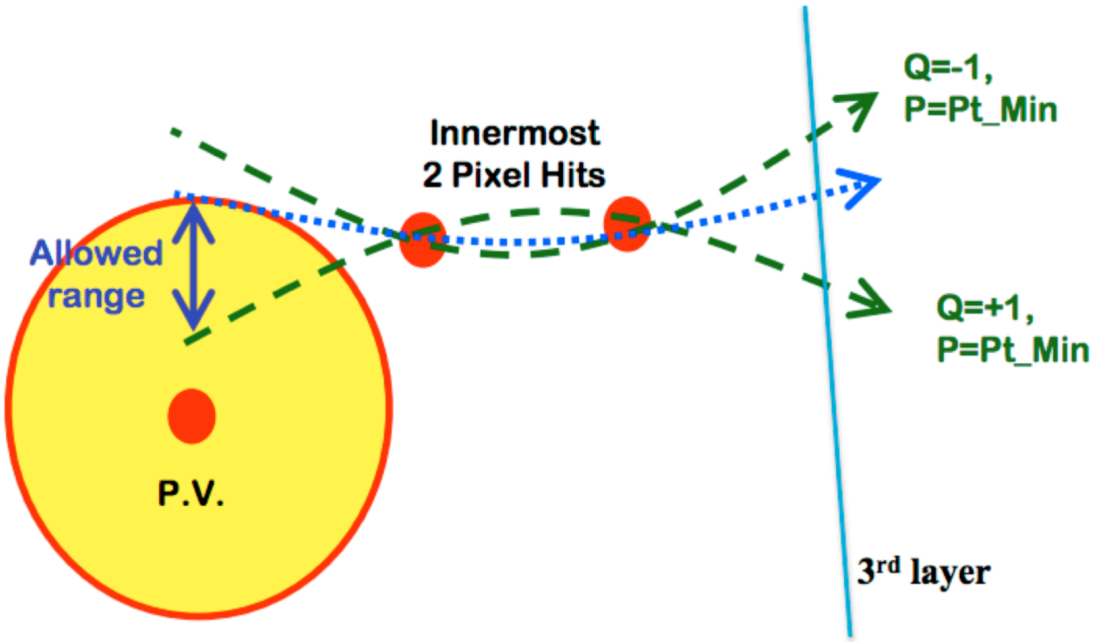
- Seeding tracks using matched hits in the pair of strip stereo layers is hard:
  - In TIB (TOB) the z-resolution is 200 (500) µm.
  - Each phi hit may be matched to several ghost hits in z (increasing as occupancy squared).
  - Since the matched layers are ~7mm apart, one must know the track angle to match the hits properly.





Seed Triplets

- After finding a hit pair, extrapolate circles of radius PtMin through the two hits for both charge hypotheses to determine the hit search window in 3rd layer.
- The allowed d0 tolerance may further restrict this window.
- In the iterative tracking two different triplet generators are used, one for prompt tracks and one for displaced tracks
  - Iterations 0-1 (prompt) use PixelTripletHLTGenerator
  - Iterations 3-4 (displaced) instead use PixelTripletLargeTipGenerator

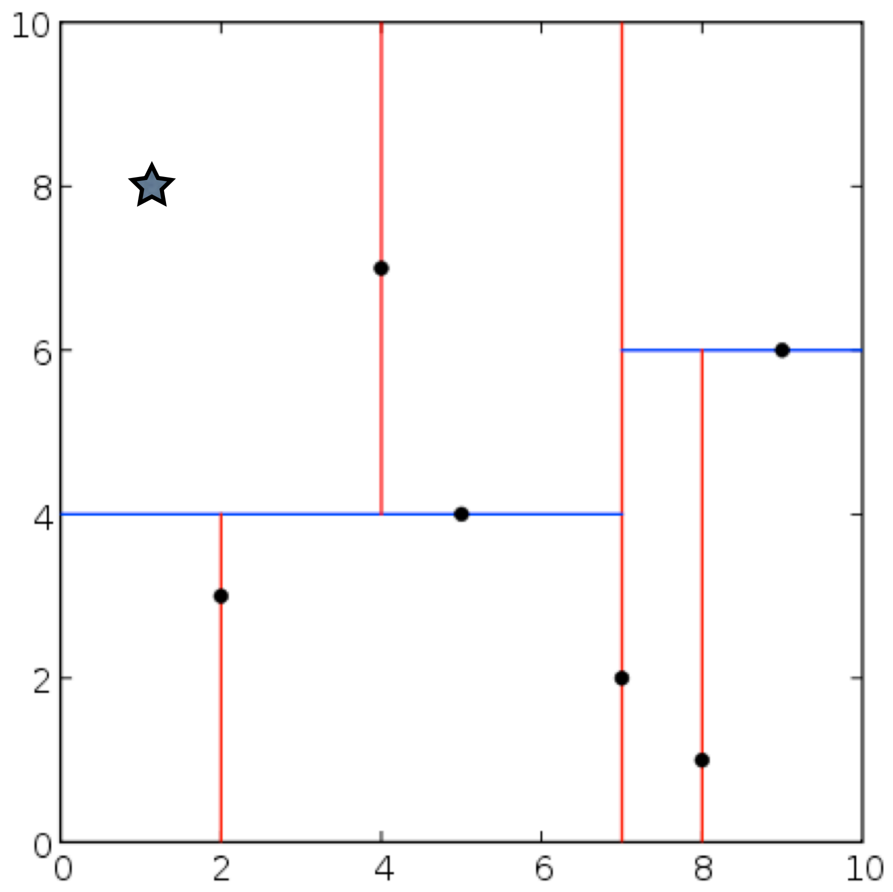


Algorithm Overview

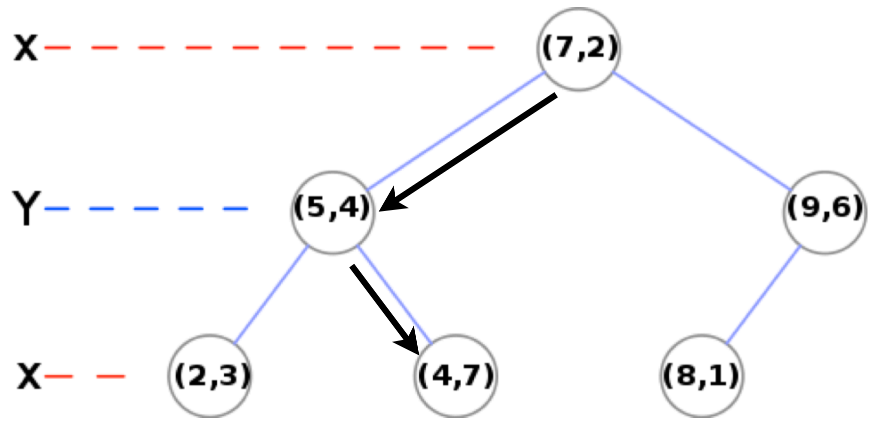
1. Layers to be used for triplets are defined in the configuration file
2. Hits in the chosen layers are sorted in phi
3. Pairs are built from hits the first two layers
    - a compatibility with the TrackingRegion is required to build the pair
4. Hits in the third layer are organized in a k-d tree in (ϕ,z) for fast search
5. For each pair, a compatible range in (ϕ,z) is defined, based on the minimum p_T allowed and a propagation in RZ plane
6. For all hits in the range returned by the k-d tree, a further cut in ϕ and z is applied to select hits to build the triplet
    - details differ in PixelTripletHLTGenerator and PixelTripletLargeTipGenerator
    - it takes into account several corrections: Multiple Scattering and bending in RZ (deviations from straight line hypothesis)
7. For each triplet, a forward kalman filter fit is performed to extract the seed parameters at the outer hit

K-D Tree

- A k-d tree is a space-partitioning data structure for organizing points in a k-dimensional space.
- k-d trees are a useful data structure for several applications, such as range searches and nearest neighbor searches.
- Algorithm used for fast search of hits in RZ (Rϕ) plane for barrel (endcap)
  - avoid looping over all layer hits when looking for the third hit of a triplet
  - significant gain in timing for seeding modules

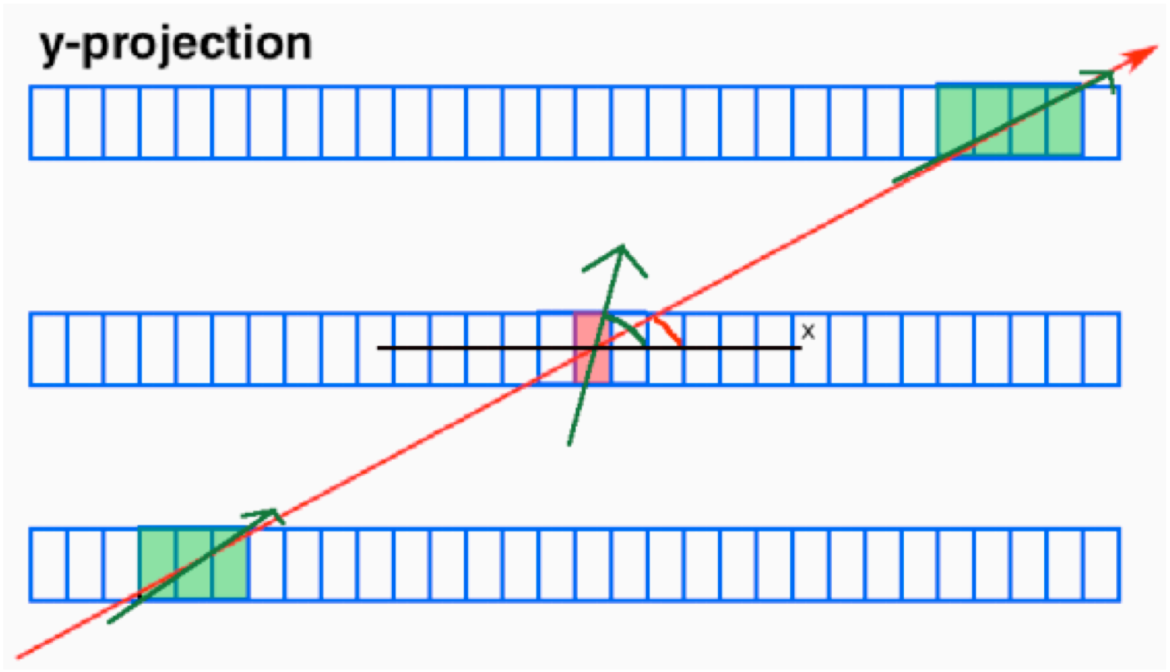


Example: find hit closest to ☆



Seed Filtering with Cluster Shape

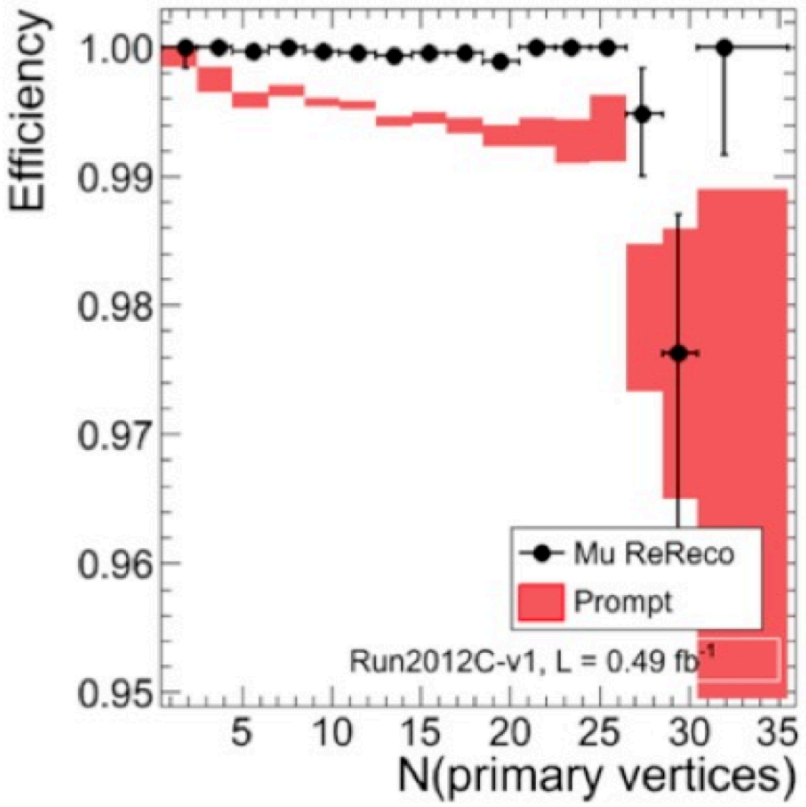
- ClusterShapeSeedComparator used by seed finder to kill seeds that contain a cluster, whose width is not compatible with that expected, given the angle at which the seed trajectory crosses the layer.



Specialized Tracking Steps

Muon Seeded Steps

- Outside-In
  - Goal: recover tracker muon efficiency for standalone mu without a tracker track
  - Start from the trajectory of the standalone muon updated at the beam line.
  - Examine the TOB & TEC layers outside-in:
    - Search for hits compatible with the muon (pick only up to 3 hits in a single layer)
    - For each hit, make a track seed using just that hit and the information from the standalone muon
    - Stop searching layers after 3 successful layers.
  - Then perform the usual CKF patern reco.
- Inside-Out
  - Goal: recover ID efficiency for global muons not passing tight selection
  - After the generalTracks reco, run tracker muon identification on tracks with $p_T > 2$ GeV.
  - Make a seed using the innermost hits of the track, up to 5 layers.
    - Always pick full layers, to avoid issues with overlaps and splitting of matched rechits.
  - Proceed with inside-out tracking with relaxed criteria, followed by seed region rebuilding



Electron and Conversion Reconstruction

- The electron reconstruction requires a collection of tracking seeds. This is currently created in the following way:
  - Collect most of the seeds found during track reconstruction
  - Use hit masking to hide hits found in these seeds (in addition to hits used by tracks)
  - Run additional seed finding using a configuration which would produce too many seeds to be used in general track finding.
- Conversion Reconstruction (ConvStep):
  - Special seeding which loops through already found tracks under the assumption that they are half of a photon conversion and tries to find the other leg.
  - In use but tracks found are only used to make conversions due to large fake rate.

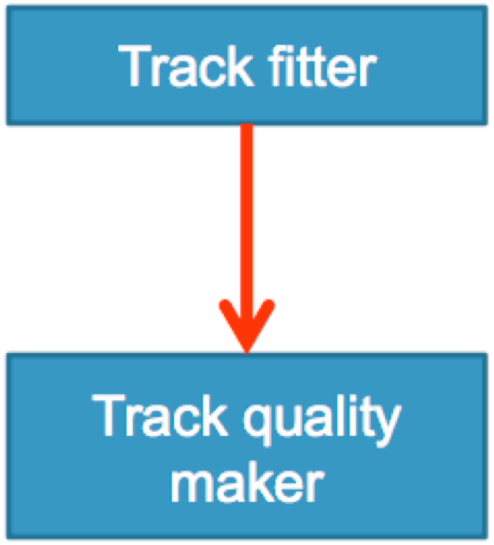
Other Tools for Iterative Tracking

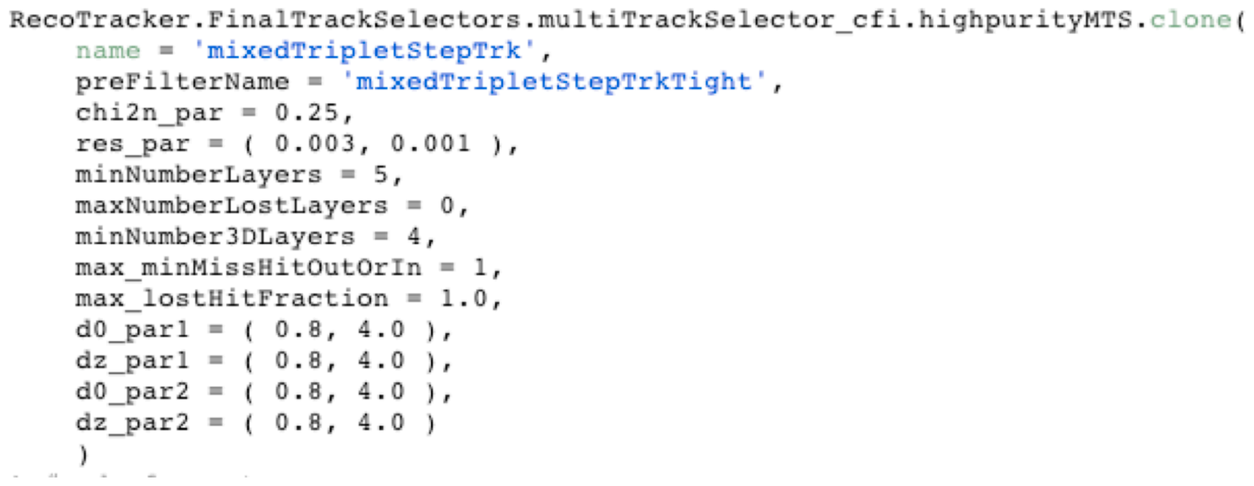
Final Track Selection

- The final track selection does the following: assigns quality flags of loose, tight, highPurity to each track and keeps only tracks passing at least loose quality. The goal is to reduce the fake rate.
- The same code MultiTrackSelector in RecoTracker/FinalTrackSelection is used to assign each quality; just different parameters. Main variables are:
  - Minimum number of layers with a hit
  - Minimum number of 3D layers with a hit
  - Maximum number of layers with a missing hit
  - χ^2/ndof
  - $d_0/\delta d_0$, $d_z/\delta d_z$, $d_0/\sigma_{d0}(p_T)$, $d_z/\sigma_{dz}(p_T,\eta)$
- with the following definitions:
  - Two hits on one layer (due to module overlaps or double-sided silicon) counts as only one layer
  - The values δd_0 and δd_z are the calculated uncertainties on d_0 and d_z which are the impact parameters to the closest pixel vertex.
  - The σ functions allow p_T (and η) dependent cuts on d_0 and d_z
    - $\sigma(d_0, z_0 \sin\theta) p(T\ ) = a + b/p_T$

Track Quality Module

- All purity assignments done in one module.
  - Output of module is value map of track qual bits
  - Filtering now done only during final merge step.
- Algorithm overview:
  - Loop over selectors (loose 1, loose 2…)
  - Loop over tracks
  - Check if track passes preselection
    - (eg, high purity must pass tight)
  - Check if track passes selection criteria
  - Put final merged set of track quality bits into ValueMap





```
RecoTracker.FinalTrackSelectors.multiTrackSelector_cfi.highpurityMTS.clone(
    name = 'mixedTripletStepTrk',
    preFilterName = 'mixedTripletStepTrkTight',
    chi2n_par = 0.25,
    res_par = ( 0.003, 0.001 ),
    minNumberLayers = 5,
    maxNumberLostLayers = 0,
    minNumber3DLayers = 4,
    max_minMissHitOutOrIn = 1,
    max_lostHitFraction = 1.0,
    d0_par1 = ( 0.8, 4.0 ),
    dz_par1 = ( 0.8, 4.0 ),
    d0_par2 = ( 0.8, 4.0 ),
    dz_par2 = ( 0.8, 4.0 )
    )
```

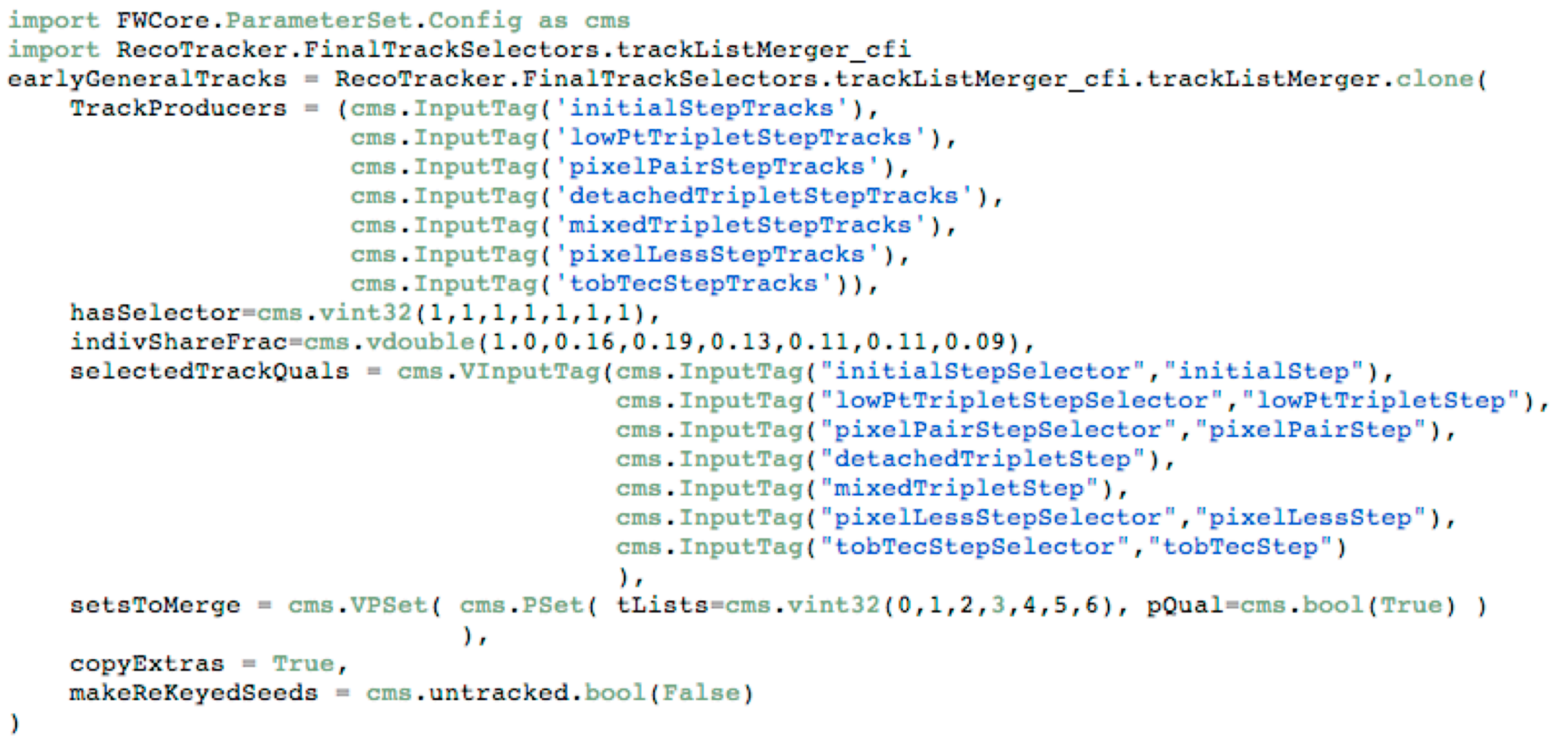
Hit Masking

- The main feature of iterative tracking is the hit masking (hiding used hits from future track iterations). The C++ code is RecoLocalTracker/SubCollectionProducers/src/TrackClusterRemover.cc
- A mask is used and passed to the track builder to identify hits which should not be used. It also handles masking by previous iterations. Hits are masked if the tracks have a given quality (highPurity), have a given number of hits (0), and the hit itself has a χ^2 less than a given value (9).
- Example call:

```
pixelLessStepClusters = cms.EDProducer("TrackClusterRemover",
    clusterLessSolution = cms.bool(True),
    oldClusterRemovalInfo = cms.InputTag("mixedTripletStepClusters"),
    trajectories = cms.InputTag("mixedTripletStepTracks"),
    overrideTrkQuals = cms.InputTag('mixedTripletStep'),
    TrackQuality = cms.string('highPurity'),
    minNumberOfLayersWithMeasBeforeFiltering = cms.int32(0),
    pixelClusters = cms.InputTag("siPixelClusters"),
    stripClusters = cms.InputTag("siStripClusters"),
    Common = cms.PSet(
        maxChi2 = cms.double(9.0)
    )
)
```

Track Merging

- Merge input track lists and quality lists to make effectively one big track list and have vectors to indicate which tracks are selected and their track quality
- Loop over tracks
  - Preselection cuts (eg, minpt)
  - Cache rechits on tracks for convenience (and performance)
  - Loop over sets of lists to merge
  - Loop over tracks in those sets
    - Loop over tracks in those sets again
    - Check that not in same set
    - Check if tracks are overlapping. If so, select one and compute corresponding changes in track quality
  - Merge track quality into tracks



```
import FWCore.ParameterSet.Config as cms
import RecoTracker.FinalTrackSelectors.trackListMerger_cfi
earlyGeneralTracks = RecoTracker.FinalTrackSelectors.trackListMerger_cfi.trackListMerger.clone(
    TrackProducers = (cms.InputTag('initialStepTracks'),
                      cms.InputTag('lowPtTripletStepTracks'),
                      cms.InputTag('pixelPairStepTracks'),
                      cms.InputTag('detachedTripletStepTracks'),
                      cms.InputTag('mixedTripletStepTracks'),
                      cms.InputTag('pixelLessStepTracks'),
                      cms.InputTag('tobTecStepTracks')),
    hasSelector=cms.vint32(1,1,1,1,1,1,1),
    indivShareFrac=cms.vdouble(1.0,0.16,0.19,0.13,0.11,0.11,0.09),
    selectedTrackQuals = cms.VInputTag(cms.InputTag("initialStepSelector","initialStep"),
                                       cms.InputTag("lowPtTripletStepSelector","lowPtTripletStep"),
                                       cms.InputTag("pixelPairStepSelector","pixelPairStep"),
                                       cms.InputTag("detachedTripletStep"),
                                       cms.InputTag("mixedTripletStep"),
                                       cms.InputTag("pixelLessStepSelector","pixelLessStep"),
                                       cms.InputTag("tobTecStepSelector","tobTecStep")
                                       ),
    setsToMerge = cms.VPSet( cms.PSet( tLists=cms.vint32(0,1,2,3,4,5,6), pQual=cms.bool(True) )
                             ),
    copyExtras = True,
    makeReKeyedSeeds = cms.untracked.bool(False)
)
```

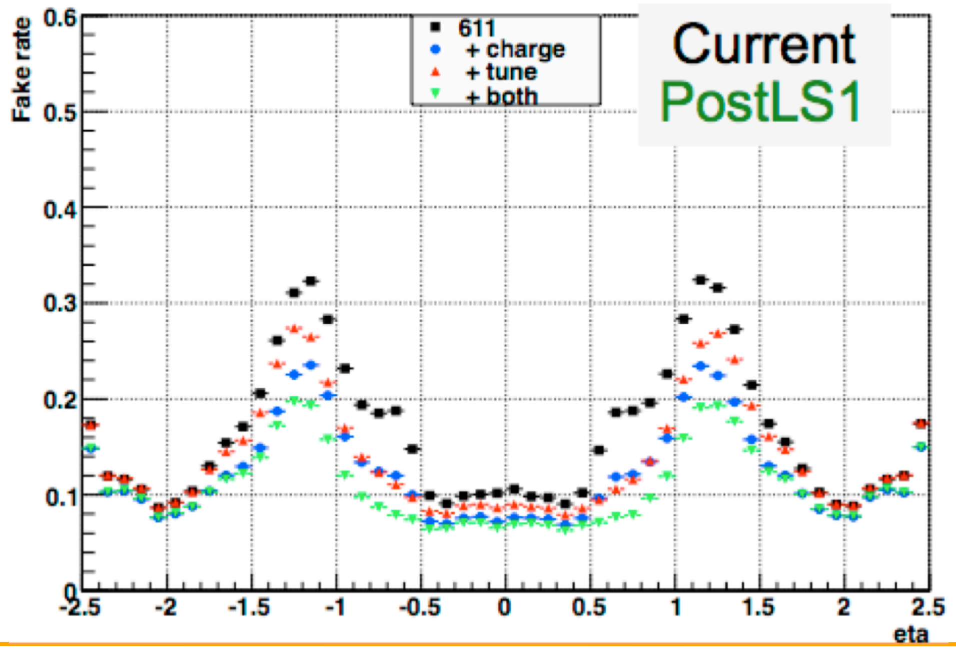
Duplicate Track Removal

- Overall, 2% of reconstructed tracks are duplicate of another track
- In general, track parameters in duplicate tracks agree fairly well
- They are identified using DuplicateTrackMerger:
  - Duplicate tracks are identified using a BDTG
    - variables used are number of missing in/outer hits and differences in track parameters
    - cut also on signed distance between outer hit of inner track and inner hit of outer track
  - Creates TrackCandidate using hits of two tracks
  - Disallows two hits in same layer from the two tracks
  - Resorts hits along momentum
- Merged TrackCandidates are fit into a new Track
  - mergedDuplicateTracks track collection
  - standard track fit
- These tracks are then merged back with all other tracks to make the generalTracks collection using DuplicateListMerger:
  - a complication may rise because of outlier rejection, stripping too many hits
  - in this case the track with best chi2 between the original tracks and the merged one is used

Recent Developments

Tracking Configurations at different PU

- Existing tracking configurations:
  - Current default: Optimized for 2012 running (20PU at 50ns bunch spacing)
  - LowPU: Optimized for low PU (2011 and dedicated low PU runs)
  - PostLS1: Optimized for 40PU and 25ns bunch crossing for 2015
  - Phase1PU70: Optimized for phase 1 detector at 70PU and 25ns bunch crossing (to be used for PU≤100)
  - Phase1PU140: Optimized for phase 1 detector at 140PU and 25ns bunch crossing (to be used for PU>100)
  - Phase2PU140: Tracking configuration for phase 2 detector at 140PU and 25ns bunch crossing
- Configurations for 25ns apply a cluster charge cut to mitigate OOTPU
  - this cut still needs substantial work before it can be used for data-taking
- PostLS1 configuration faster, lower fake rate, small efficiency loss
  - loss mainly for low p_T displaced tracks



Conf/Iteration	Current	PostLS1
Initial	1.58 s	1.51 s
LowPt	1.57 s	1.50 s
PixelPair	1.61 s	1.53 s
DetachedTrip	0.84 s	0.82 s
MixedTrip	1.75 s	1.33 s
PixelLess	11.50 s	2.21 s
TobTec	8.74 s	0.82 s
Total time	**29.13 s**	**10.85 s**

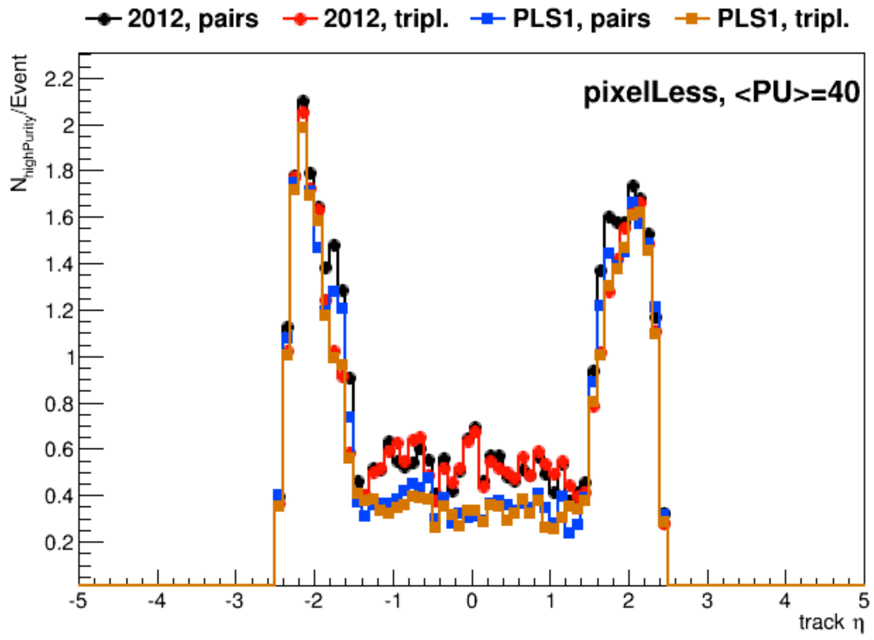
Current vs PostLS1 tracking

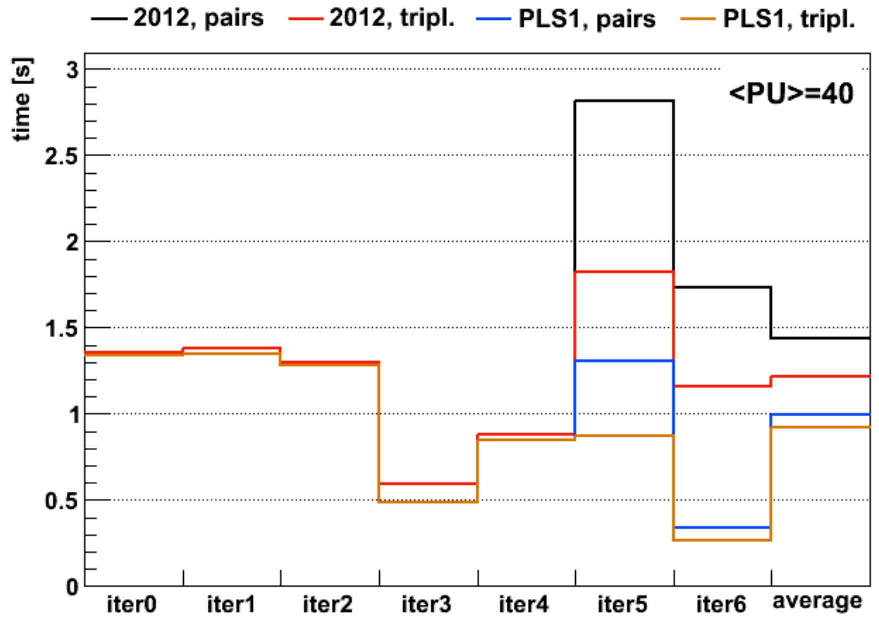
	$O_Z/2$ [cm]	O_R [cm]	p_T pair [GeV]	p_T triplets [GeV]
pixelLess 2012	15.0	2.5	0.7	0.4
pixelLess PLS1	12.0 - 15.0 (*)	1.0 - 1.5 (*)	0.9 - 0.7 (*)	0.55 - 0.4 (*)
tobTec 2012	30.0	6.0	0.6	0.5 - 0.6 (**)
tobTec PLS1	20.0	3.5	0.8	0.65 - 0.8 (**)

(*): first number is for barrel, second for endcap
(**): first number is for barrel (triplets), second for endcap (pairs)

Strip Triplet Seeding

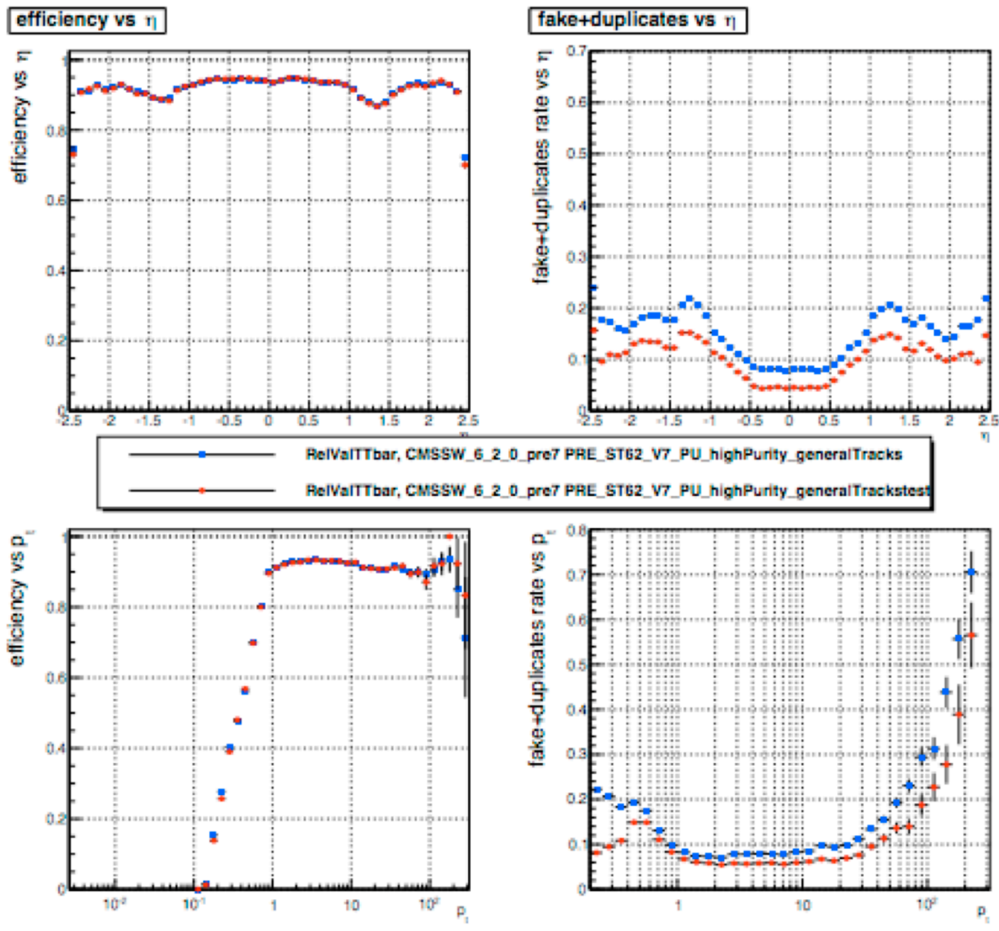
- A strip triplet seeding (instead of strip pair) is effective in reducing the timing spent during the pattern recognition for iterations 5 and 6.
- The current triplet-making algorithm can be simplified by removing time consuming corrections and by applying a chi2 cut computed from the linear fit of the 3 hits in the RZ plane.
- Triplet-based timing shows: <2x increase for seeding time, >2x decrease for building time.
- Total time gain is 35% for pixelLess step, small efficiency loss for low pT displaced tracks
- Combining PostLS1 configurations and triplet-based seeding, timing for pixelLess and tobTec steps is reduced by 3x and 6x respectively.





MVA Track Selection

- Improve the fake rate and real track efficiency by classifying them using a multivariate approach
- Then use the MVA when evaluating the quality tags (loose, highPurity)
- Trained separately for each iteration using BDTG in TMVA
- Variables:
  - Number of layers
  - Number of 3D layers
  - Number of layers lost
  - Chi2n
  - Chi2n no1D modification*
  - Eta*
  - Relative pT error*
  - Number of hits*
  - Minimum lost hits from outer or inner
  - Fraction of Lost Hits*
    - where * means that this variable is not used in current cut-based selection.
    - dZ and d0 are used in cut based but not in MVA



Tracking at HLT
(thanks to A.Tropiano for the help)

Usage of tracking at HLT

- To limit CPU usage at HLT several things are slightly different:
  - General
    - tracking is done on demand.
    - tracking is done in regions of interest.
  - Primary Vertex
    - 1D “divisive” vertex finder still used at HLT (but planning to use DA in 2015).
    - pixel Tracks are used to build the vertex.
  - Track Fitting
    - don’t split hits in double sided modules.
- Different tracking configurations run depending on HLT path:
  - Jet, MET, HT reconstruction
    - PFJets reconstruction uses iterative tracking at HLT.
    - PU subtraction relies on iterative tracking too.
  - Lepton reconstruction isolation
    - pixelTracks match used for electron reconstrucion.
    - inside out tracking for muons.
    - PF isolation not used yet. Regional tracks are used for detector based isolation.
  - b-tag
    - pixelTracks used to reduce timing (L2.5 btagging).
    - full tracks used to cut on final discriminant (L3 btagging).
    - Secondary Vertex algorithm also used (adaptive vertexing).

Iterative Tracking at HLT

HLT (offline) cuts

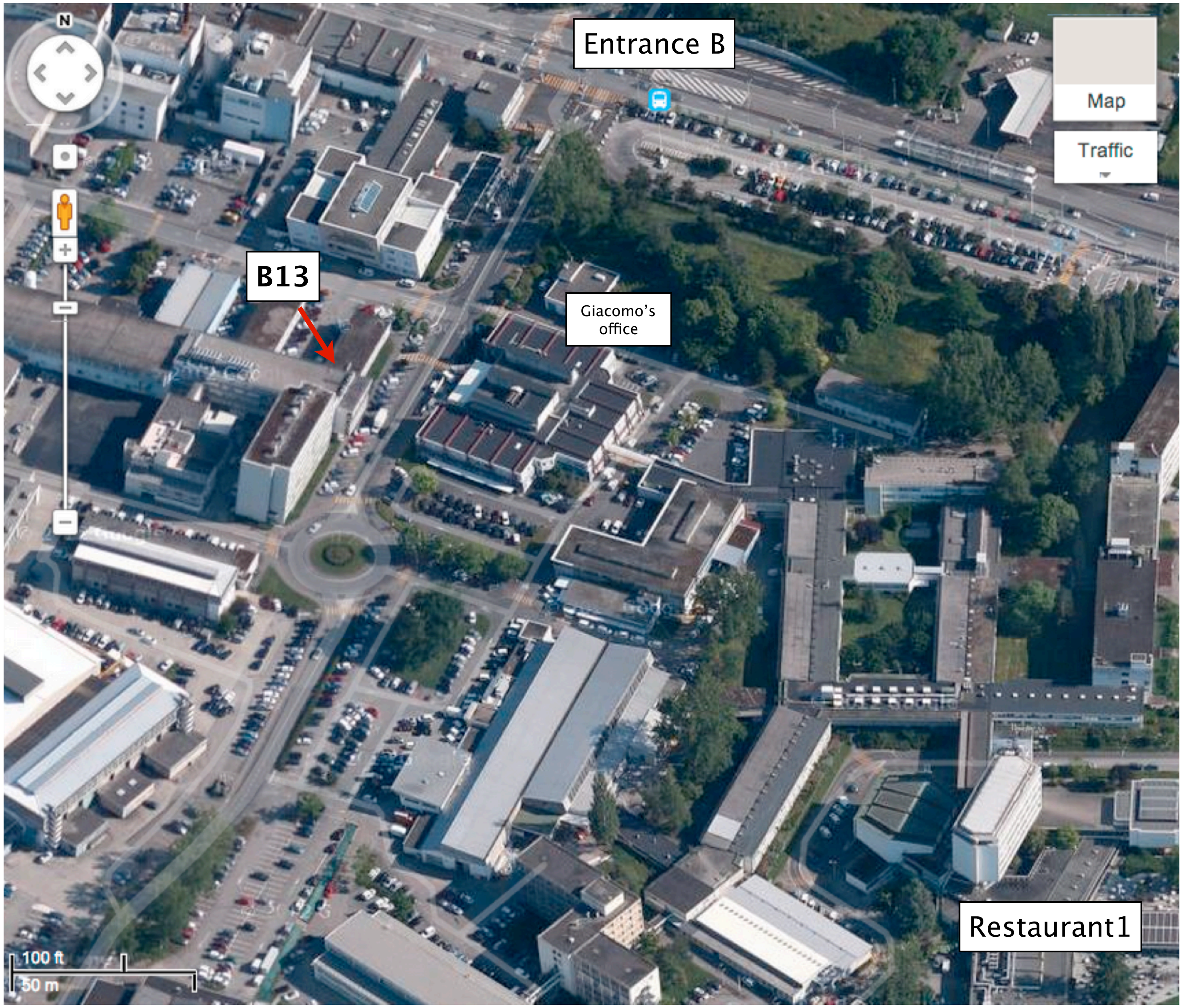
Iteration	Seeds	pT cut [GeV]	d0 cut [cm]	dz cut
0	pixel triplets	0.8(0.6)	0.1(0.02)	0.3cm(4.0σ)
1	pixel triplets	0.5(0.2)	0.05(0.02)	0.1cm(4.0σ)
2	pixel pairs	1.2(0.6)	0.025(0.015)	0.05cm(4.0σ)
3 (3-4)	pixel,TIB,TID,TEC	0.8(0.3)	0.05(1.5)	0.05cm(10.0cm)
4 (5)	TIB,TID,TEC	0.8(0.7)	0.5(2.5)	0.5cm(15.0cm)
- (6)	TOB,TEC	NA(0.6)	NA(6.0)	NA(30.0cm)

This keeps the tracking running time lower than 1s at HLT (2012 data).

(Incomplete list of) References

- Tracking Paper in CWR:
  - http://cms.cern.ch/iCMS/analysisadmin/paperversions?analysis=TRK-11-001
- Talk on Duplicate Merging:
  - Walker: https://indico.cern.ch/getFile.py/access?contribId=3&resId=0&materialId=slides&confId=211273
- Talk on Track Merging:
  - Lange: https://indico.cern.ch/getFile.py/access?contribId=17&resId=1&materialId=slides&confId=186022
- Talk on Iterative Tracking:
  - Stenson: https://indico.cern.ch/getFile.py/access?contribId=13&resId=0&materialId=slides&confId=186022
- Talk on Seeding:
  - Tomalin: https://indico.cern.ch/getFile.py/access?contribId=7&resId=0&materialId=slides&confId=186022
- Talk on Muon Seeded Steps:
  - Petrucciani: https://indico.cern.ch/getFile.py/access?contribId=5&resId=1&materialId=slides&confId=203694
- Talk on Iterative Tracking Configurations at different PU:
  - Stenson: https://indico.cern.ch/getFile.py/access?contribId=1&resId=0&materialId=slides&confId=261789
- Talk on Strip Triplet Seeding
  - Cerati: https://indico.cern.ch/getFile.py/access?contribId=4&resId=0&materialId=slides&confId=261789
- Talk on MVA Track Selection
  - Walker: https://indico.cern.ch/getFile.py/access?contribId=4&resId=0&materialId=slides&confId=271312

See you at 14:00 in room 13-2-005 for the hands-on session!


Entrance B
Map
Traffic
B13
Giacomo's office
Restaurant1
100 ft
50 m