# Table of Contents

# Iterative Tracking

Complete:

## Goal of the page

The reader of the page should become familiar with the iterative tracking approach, and be able to add tracking steps in the whole tracking chain.

## Contacts

Michele Pioppi: michele.pioppi@cernNOSPAMPLEASE.ch
Kevin Stenson: kevin.stenson@coloradoNOSPAMPLEASE.edu

## Introduction

The iterative tracking approach runs the standard CTF tracking algorithm multiple times. At each iteration, the hits used by previous iterations are removed from consideration and the CTF tracking algorithm is run again with progressively looser settings. The presentations below trace the development of the iterative tracking approach:
Tracking meeting 23 may 2007
PFlow meeting 05 july 2007
Tracking meeting 01 august 2007
CMS.ParticleFlow meeting 13 september 2007
Tracking meeting 31 january 2008
Tracking meeting 05 february 2008
Tracking meeting 14 february 2008
The results of the iterative tracking are summarized in the internal note 2007/065

### Summary of the current iterative steps

Show CMSSW42X information Hide CMSSW42X information
The 4_2_X default tracking in CMS contains 6 iterations, labeled 0 through 5. The main distinction between the iterations is the track seeding algorithm which is shown in the table below.

| Iteration | Seeding Layers | pT cut (GeV) | d0 cut (cm) | z0 cut |
|---|---|---|---|---|
| Zero | pixel triplets | 0.8 | 0.2 | 3.0 |
| 1 | pixel pairs | 0.6 | 0.05 | 0.2cm* |
| 2 | pixel triplets | 0.075 | 0.2 | 3.3 |
| 3 | Triplets: pixel, TIB1,2, TID/TEC ring 1,2 | 0.25-0.35 | 2.0 | 10.0 |
| 4 | Pairs: TIB1,2 & TID/TEC ring 1,2 | 0.5 | 2.0 | 12.0 |
| 5 | Pairs: TOB1,2 & TEC ring 5 | 0.6 | 6.0 | 30.0 |

In the table, d0 and z0 refer to the transverse and longitudinal impact parameters of seeds with respect to the nominal interaction point. The * indicates the impact parameter with respect to a pixel vertex. Using triplet seeding is much faster and has a lower fake rate than pairs. Therefore, pixel triplet seeding is run first (iteration 0), followed by pixel pairs (iteration 1) for additional efficiency. Iteration 2 uses pixel triplets like iteration 0 but searches for very low momentum tracks. Iteration 3 uses pixels and strip triplets to find tracks which miss a pixel layer and also tracks which may decay within a couple of cm of the production vertex. Iterations 4 and 5 do not use pixels to seed and are designed to find tracks which are significantly displaced from the beam line or tracks which do not leave sufficient pixel hits to be found in the earlier iterations. Other differences between iterations during track building include the minimum number of hits (3 for iterations 0-3, 6 for iteration 4, and 6 for iterations 4-5), the number of lost hits (1 for iterations 0-2 and 0 for iterations 3-5).

The final cleaning stage is also different. The early steps have stricter requirements on tracks originating from the production vertex while the later steps have stricter requirements on the track quality. Details will eventually be found in the tracking note but for now they can be found in the Very Large Impact Parameter Track Reconstruction note or by looking in the configuration steps which are linked in the table above and annotated more below.

# Usage of the algorithm

In each tracking iteration the user must:

- Create a new cluster collection by removing the clusters used in the previous track collections

e.g.

```
firstfilter = cms.EDFilter("QualityFilter",  # Selecting tracks from the previous iteration
    TrackQuality = cms.string('highPurity'),  # Just take the highPurity ones
    recTracks = cms.InputTag("preMergingFirstStepTracksWithQuality")
)
 secClusters = cms.EDFilter("TrackClusterRemover",
    oldClusterRemovalInfo = cms.InputTag("newClusters"), # clusters used in the previous iteratio
    trajectories = cms.InputTag("firstfilter"),    # trajectories of the previous iteration
    pixelClusters = cms.InputTag("newClusters"), # pixel clusters used in the previous iteration
    stripClusters = cms.InputTag("newClusters"), # strip clusters used in the previous iteration
    Common = cms.PSet(
        maxChi2 = cms.double(30.0)
    )
)
```

- Create new pixel and strip collections with the new cluster collection

e.g.

```
import CMS.RecoLocalTracker.SiPixelRecHits.SiPixelRecHits_cfi
secPixelRecHits = CMS.RecoLocalTracker.SiPixelRecHits.SiPixelRecHits_cfi.siPixelRecHits.clone(
    src = 'secClusters'
    )
import CMS.RecoLocalTracker.SiStripRecHitConverter.SiStripRecHitConverter_cfi
secStripRecHits = CMS.RecoLocalTracker.SiStripRecHitConverter.SiStripRecHitConverter_cfi.siStripM
    ClusterProducer = 'secClusters'
    )
```

- Run the new tracking algorithm with the new cluster and hit collections.
- The new hit and cluster collection must be put in the seeding

e.g.

```
import RecoTracker.TkSeedingLayers.PixelLayerTriplets_cfi
seclayertriplets = RecoTracker.TkSeedingLayers.PixelLayerTriplets_cfi.pixellayertriplets.clone(
    ComponentName = 'SecLayerTriplets'
    )
seclayertriplets.BPix.HitProducer = 'secPixelRecHits'
seclayertriplets.FPix.HitProducer = 'secPixelRecHits'
```

- in the measurement tracker

e.g.

```
import RecoTracker.MeasurementDet.MeasurementTrackerESProducer_cfi
secMeasurementTracker = RecoTracker.MeasurementDet.MeasurementTrackerESProducer_cfi.MeasurementTr
    ComponentName = 'secMeasurementTracker',
    pixelClusterProducer = 'secClusters',
    stripClusterProducer = 'secClusters'
    )
```

- and in the fitter

e.g.

```
import RecoTracker.TrackProducer.TrackProducer_cfi
secWithMaterialTracks = RecoTracker.TrackProducer.TrackProducer_cfi.TrackProducer.clone(
    AlgorithmName = cms.string('iter2'),
    src = 'secTrackCandidates',
    clusterRemovalInfo = 'secClusters'
    )
```

Show CMSSW44X information Hide CMSSW44X information

The 4_4_X default tracking in CMS contains 7 iterations, labeled 0 through 6. These correspond to algo values 4-10. The main distinction between the iterations is the track seeding algorithm which is shown in the table below.

| Iteration | Seeding Layers | pT cut (GeV) | d0 cut (cm) | z0 cut |
|---|---|---|---|---|
| InitialStep (0) | pixel triplets | 0.6 | 0.03 | 4.0 |
| LowPtTripletStep (1) | pixel triplets | 0.2 | 0.03 | 4.0 |
| PixelPairStep (2) | pixel pairs | 0.6 | 0.01 | 0.09cm* |
| DetachedTripletStep (3) | pixel triplets | 0.2 | 1.0 | 4 |
| MixedTripletStep (4) | Triplets: pixel, TIB1,2, TEC ring 1,2, wheels 1-3 | 0.35-0.5 | 2.0 | 10.0 |
| PixelLessStep (5) | Pairs: TIB1,2 & TID/TEC ring 1,2 | 0.6 | 2.0 | 10.0 |
| TobTecStep (6) | Pairs: TOB1,2 & TEC ring 5, wheels 1-7 | 0.6 | 6.0 | 30.0 |

The 5_X_X default tracking in CMS contains 7 iterations, labeled 0 through 6. These correspond to algo values 4-10. The main distinction between the iterations is the track seeding algorithm which is shown in the table below.

| Iteration | Seeding Layers | pT cut (GeV) | d0 cut (cm) | z0 cut |
|---|---|---|---|---|
| InitialStep (0) | pixel triplets | 0.6 | 0.02 | 4.0 |
| LowPtTripletStep (1) | pixel triplets | 0.2 | 0.02 | 4.0 |
| PixelPairStep (2) | pixel pairs | 0.6 | 0.015 | 0.09cm* |
| DetachedTripletStep (3) | pixel triplets | 0.3 | 1.5 | 15cm |
| MixedTripletStep (4) | Triplets: pixel, TIB1,2, TEC ring 1,2, wheels 1-3 | 0.4-0.6 | 1.5 | 10cm |
| PixelLessStep (5) | Pairs: TIB1,2 & TID/TEC ring 1,2 | 0.7 | 2.0 | 10cm |
| TobTecStep (6) | Pairs: TOB1,2 & TEC ring 5, wheels 1-7 | 0.6 | 6.0 | 30cm |

In the table, d0 and z0 refer to the transverse and longitudinal impact parameters of seeds with respect to the beamspot. The * indicates the impact parameter with respect to a pixel vertex. Using triplet seeding is much faster and has a lower fake rate than pairs. Therefore, pixel triplet seeding is run first for higher pT (iteration 0) and lower pT (iteration 1), followed by pixel pairs (iteration 2) for additional efficiency. Iteration 3 uses pixel triplets like iterations 0,1 but searches for displaced tracks. Iteration 4 uses pixels and strip triplets to find tracks which miss a pixel layer and also tracks which may decay within a couple of cm of the production vertex. Iterations 4 and 5 do not use pixels to seed and are designed to find tracks which are significantly displaced from the beam line or tracks which do not leave sufficient pixel hits to be found in the earlier iterations. Other differences between iterations during track building include the minimum number of hits (3 for iterations 0-4 and 6 for iterations 5-6), the number of lost hits (1 for iterations 0-3 and 0 for iterations 4-6).

The final cleaning stage is also different. The early steps have stricter requirements on tracks originating from the production vertex while the later steps have stricter requirements on the track quality. Details can be found by looking in the configuration steps which are linked in the table above and annotated more below.

# Usage of the algorithm

In each tracking iteration the user must:

- Identify the clusters used in the previous track collections such as:

```
mixedTripletStepClusters = cms.EDProducer("TrackClusterRemover",
    clusterLessSolution = cms.bool(True),
    oldClusterRemovalInfo = cms.InputTag("detachedTripletStepClusters"),
    trajectories = cms.InputTag("detachedTripletStepTracks"),
    overrideTrkQuals = cms.InputTag('detachedTripletStep'),
    TrackQuality = cms.string('highPurity'),
    pixelClusters = cms.InputTag("siPixelClusters"),
    stripClusters = cms.InputTag("siStripClusters"),
    Common = cms.PSet(
        maxChi2 = cms.double(9.0)
    )
)
```

- Input this information at the seeding level. This is done by the skipClusters command in the seed layers definition below:

```
mixedTripletStepSeedLayersB = cms.ESProducer("SeedingLayersESProducer",
    ComponentName = cms.string('mixedTripletStepSeedLayersB'),
    layerList = cms.vstring('BPix2+BPix3+TIB1',
        'BPix2+BPix3+TIB2','BPix3+TIB1+TIB2'),
    BPix = cms.PSet(
        useErrorsFromParam = cms.bool(True),
        hitErrorRPhi = cms.double(0.0027),
        hitErrorRZ = cms.double(0.006),
        TTRHBuilder = cms.string('TTRHBuilderWithoutAngle4MixedTriplets'),
        HitProducer = cms.string('siPixelRecHits'),
        skipClusters = cms.InputTag('mixedTripletStepClusters')
    ),
    TIB = cms.PSet(
        matchedRecHits = cms.InputTag("siStripMatchedRecHits","matchedRecHit"),
        TTRHBuilder = cms.string('WithTrackAngle'),
        skipClusters = cms.InputTag('mixedTripletStepClusters')
    )
)
```

- The trajectory builder needs to be informed with the "clustersToSkip" parameter as shown below:

```
import RecoTracker.CkfPattern.GroupedCkfTrajectoryBuilderESProducer_cfi
mixedTripletStepTrajectoryBuilder = RecoTracker.CkfPattern.GroupedCkfTrajectoryBuilderESProducer_
    ComponentName = 'mixedTripletStepTrajectoryBuilder',
    MeasurementTrackerName = '',
    trajectoryFilterName = 'mixedTripletStepTrajectoryFilter',
    propagatorAlong = cms.string('mixedTripletStepPropagator'),
    propagatorOpposite = cms.string('mixedTripletStepPropagatorOpposite'),
    clustersToSkip = cms.InputTag('mixedTripletStepClusters'),
    maxCand = 2,
    estimator = cms.string('mixedTripletStepChi2Est')
    )
```

- The track candidate maker needs the trajectory builder name and the track producer needs the track candidate name:

```
import RecoTracker.CkfPattern.CkfTrackCandidates_cfi
mixedTripletStepTrackCandidates = RecoTracker.CkfPattern.CkfTrackCandidates_cfi.ckfTrackCandidate
    src = cms.InputTag('mixedTripletStepSeeds'),
    TrajectoryBuilder = 'mixedTripletStepTrajectoryBuilder',
    doSeedingRegionRebuilding = True,
    useHitsSplitting = True
)
import RecoTracker.TrackProducer.TrackProducer_cfi
mixedTripletStepTracks = RecoTracker.TrackProducer.TrackProducer_cfi.TrackProducer.clone(
    AlgorithmName = cms.string('iter4'),
    src = 'mixedTripletStepTrackCandidates'
)
```

# Software architecture

The code for the hit removal is in the package RecoLocalTracker/SubCollectionProducers
Very good examples for the iterative tracking usage can be found in RecoTracker/IterativeTracking

# Review status

| Reviewer/Editor and Date (copy from screen) | Comments |
|---|---|
| KatiLassilaPerini - 29 Jan 2008 | created template page |
| KevinStenson - 06 Aug 2009 | Added information on current iterative tracking steps |
| KevinStenson - 01 Oct 2011 | Updated to reflect iterative tracking CMSSW440 |

Responsible: KevinStenson
Last reviewed by: Most recent reviewer

---

This topic: CMSPublic > SWGuideIterativeTracking
Topic revision: r9 - 2012-11-01 - KevinStenson