Table of Contents

HLT b-tagging for hadronic top analysis

How it works

First of all there is a filter on L1 seed, for now we use QuadJet8U. It will be replaced by a QuadCenJet20 (both raising the threshold and dropping forward jet) to reduce the rate and still have a high efficiency. Than the reco jet sequence is ran and we ask for 4 jets with pt>25GeV/c (uncorrected energy) in the central region $|\ |<3$.

```
process.hlt4jet25U = cms.EDFilter( "HLT1CaloJet",
                                   inputTag = cms.InputTag( "hltMCJetCorJetIcone5HF07" ),
                                   saveTag = cms.untracked.bool( True ),
                                   MinPt = cms.double( 25.0 ),
                                   MaxEta = cms.double( 3.0 ),
                                   MinN = cms.int32( 4 )
                                   )
```

At this point we run the L2.5 (pixel only) and L3 (full regional tracking) b-tagging sequence on the first 4 jets of the event and ask for at least one tagged jet.

Pixel tracking and tagging (L2.5 Filter)

First of all we reconstruct and associate associate pixel tracks to jets matching them with a cone. This is done reconstructing tracks from pixel triplets and pixel vertices using the online beamspot as a reference. In particular the pixel vertex is computed using the DivisiveVertexFinder algorithm and it's a 1D method. This is the code used now:

```
process.hltPixelVertices = cms.EDProducer( "PixelVertexProducer",
    Verbosity = cms.int32( 0 ),
    Finder = cms.string( "DivisiveVertexFinder" ),
    UseError = cms.bool( True ),
    WtAverage = cms.bool( True ),
    ZOffset = cms.double( 5.0 ),
    ZSeparation = cms.double( 0.05 ),
    NTrkMin = cms.int32( 2 ),
    PtMin = cms.double( 1.0 ),
    TrackCollection = cms.InputTag( "hltPixelTracks" ),
    beamSpot = cms.InputTag( "hltOnlineBeamSpot" ),
    Method2 = cms.bool( True )
```

Performances can be improved if one uses the 3D vertex reconstruction. here is an example of how it can be implemented:

```
process.hltPixelVertices3D = cms.EDProducer("PrimaryVertexProducer",
    PVSelParameters = cms.PSet(
        maxDistanceToBeam = cms.double(2) ## 2cm
    ),
    verbose = cms.untracked.bool(False),
    algorithm = cms.string('AdaptiveVertexFitter'),
    minNdof  = cms.double(0.0),
    TkFilterParameters = cms.PSet(
        algorithm=cms.string('filter'),
        maxNormalizedChi2 = cms.double(100.0),     #
        minSiliconLayersWithHits = cms.int32(2), # none
        minPixelLayersWithHits = cms.int32(2),   # >= 2
        maxD0Significance = cms.double(100.0),     # keep most primary tracks
        minPt = cms.double(0.0),                 # better for softish events
        trackQuality = cms.string("any")
    ),
    beamSpotLabel = cms.InputTag("hltOnlineBeamSpot"),
```

```
    # label of tracks to be used
    TrackLabel = cms.InputTag("hltPixelTracks"),
    useBeamConstraint = cms.bool(False),
    # clustering
    TkClusParameters = cms.PSet(
        algorithm   = cms.string('gap'),
        TkGapClusParameters = cms.PSet(
            zSeparation = cms.double(0.2)
        )
    )
)
```

Then we compute the discriminator according to the algorithm we choose, i.e. using the 3rd or the 2nd track in the jet. In the end we filter the event requiring a minimum value for the discirminator to tag the jet (tipically 2).

Full tracking and tagging (L2.5 Filter)

The main parameters of interest to compute the tag are in the trackIPproducer hltBLifetimeL3(L25)TagInfosStartupU and in the jettagproducer hltBLifetimeL3BJetTagsStartupU, where you can choose the minimum number of hits, the minimum number of pixel hits, the track to use to compute the IP significance and so on.

Last thing. The .py to run on data misses the filters on the IP at L25 and L3. You can copy paste them (hltBLifetimeL3(25)FilterStartupU) from the other file.

Recipe for reproducing the b-tagging at HLT with v9.4 HLT menu

To reproduce the IP based b-tagging at the HLT level with this menu we just have to follow these steps:

- `cmsrel CMSSW_3_9_0`
- `cvs co HLTrigger/HLTfilters`
- `cvs co HLTrigger/JetMET`
- `cvs co -r V00-07-00 TrackingTools/Producers`
- `scram b`
- run the cfg at the following path:
  `/afs/cern.ch/user/t/tropiano/public/btagHLT/offline_data.py`

Timing studies

To run the official timing studies software one should first produce a skimmed rootuple, with just the `FEDRawDataCollection` stuff.

You can find it here: `/afs/cern.ch/user/t/tropiano/public/btagHLT/TimingSkim.py`

To-do list

- Run OpenHLT emulation on PU MinBias samples. Estimate the rate.
- Save the vertex number in Ntuple. Estimate efficiency and rate VS number of vertices.
-

The 3d vertexing is a more complex thing, there are many parameters to tune and it should be followed by one person. Also the timing is not yet clear and it is to be evaluated. Also a good thing woud be to write a code t estimate tthe efficiency of the btag pats at HLT. This would be similar to the code made for the turn on curves. It should match the HLT-offline jets. Than calculate the taggers for HLT and offline jets. Than

compare the 2 and fill some histograms of the efficiency of the algorithm, the purity and so on. Also other suggestions were made which don't make use of the 3d vertexing at HLT. One solution was to look at the 1st track in the jets. Another is to request 2 b-tags to lowere the rate. I am not sure of this and in case it can be used with the PentaJet, beacuse there are omre jets to tag and the efficiency should be higher.

-- AntonioTropiano - 30-Oct-2010

This topic: Main > BtagHLT
Topic revision: r8 - 2011-01-26 - AntonioTropiano