Table of Contents

Fake Rate Definitions

High Pt Photon ID

Our photon identification variables and cuts use the High pT photon ID V2 working points.

Sigma_IphiIphi cut

In addition to the below object definitions, we apply a cut on sigma_IphiIphi to remove beam halo. This cut is applied offline as follows

```
if (Photon_sigmaIphiIphi5x5 < 0.009) continue;
```

See our discussion of beam halo rejection on the fake rate twiki: Rejection of Beam Halo in Jet Data Sample

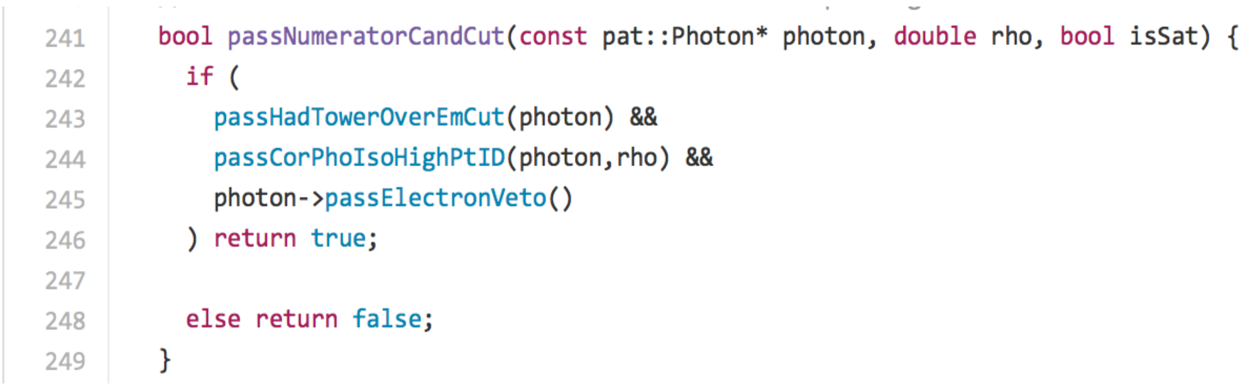
Numerator object definition

We form a "relaxed numerator" compared to what we would get solely by using our photon ID. Since we will start by using sigma_iEtaiEta templates, this is the variable we choose to relax. Our numerator definition is the same ID as the High pT photon ID, but without any sigma_iEtaiEta cut.

In the code, we select numerator object candidates

```
photonInfo.isNumeratorObjCand =
ExoDiPhotons::passNumeratorCandCut(photon,rho,isSat);
```

where



```
bool passNumeratorCandCut(const pat::Photon* photon, double rho, bool isSat) {
  if (
    passHadTowerOverEmCut(photon) &&
    passCorPhoIsoHighPtID(photon,rho) &&
    photon->passElectronVeto()
  ) return true;

  else return false;
}
```

Finally, we select our numerator objects offline as

```
bool isNumeratorObj = Photon_isNumeratorObjCand && Photon_passChIso;
```

Denominator object definition

We form a "loose-but-not-tight" denominator. An object must pass

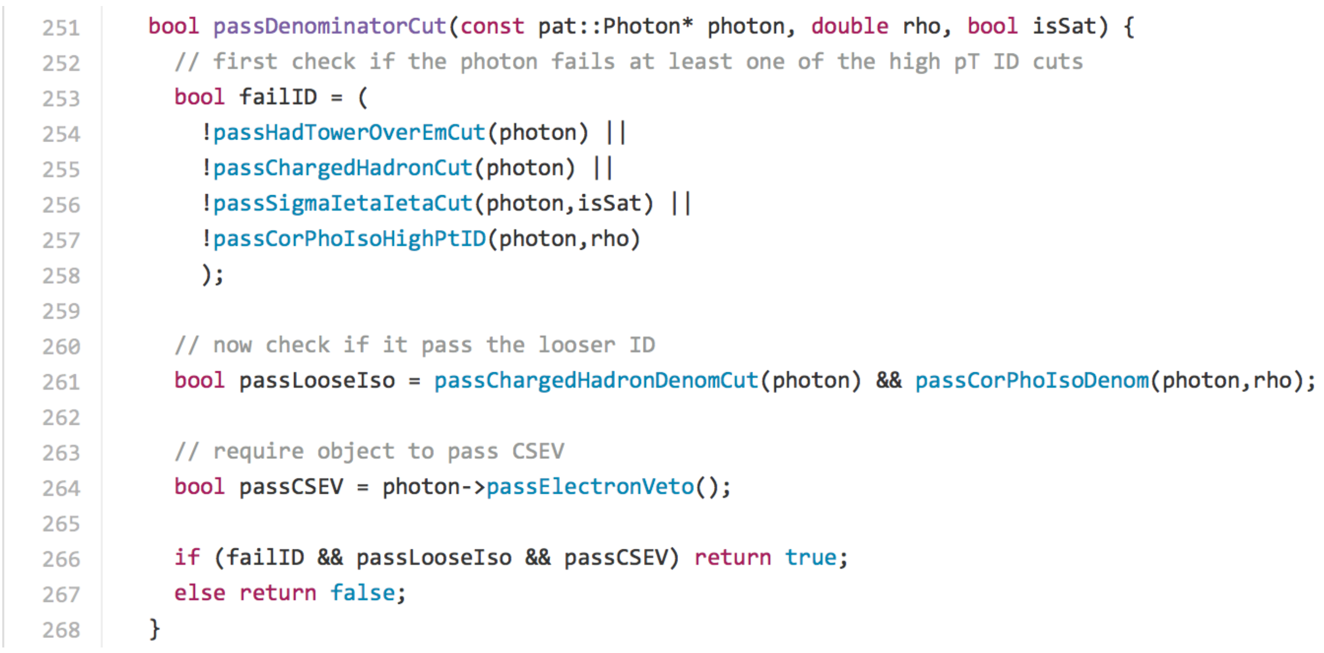
iso < min[5*(iso cut), 0.2*photon->pt()]

where "iso" is each isolation variable used in the High pT photon ID, of which there are two: charged hadron isolation and "corrected" photon isolation. In addition, we require the object to fail at least one of the High pT photon ID variables, except for the CSEV, which we require the object to pass.

In the code, we select denominator objects as

photonInfo.isDenominatorObj = ExoDiPhotons::passDenominatorCut(photon,rho,isSat);

where



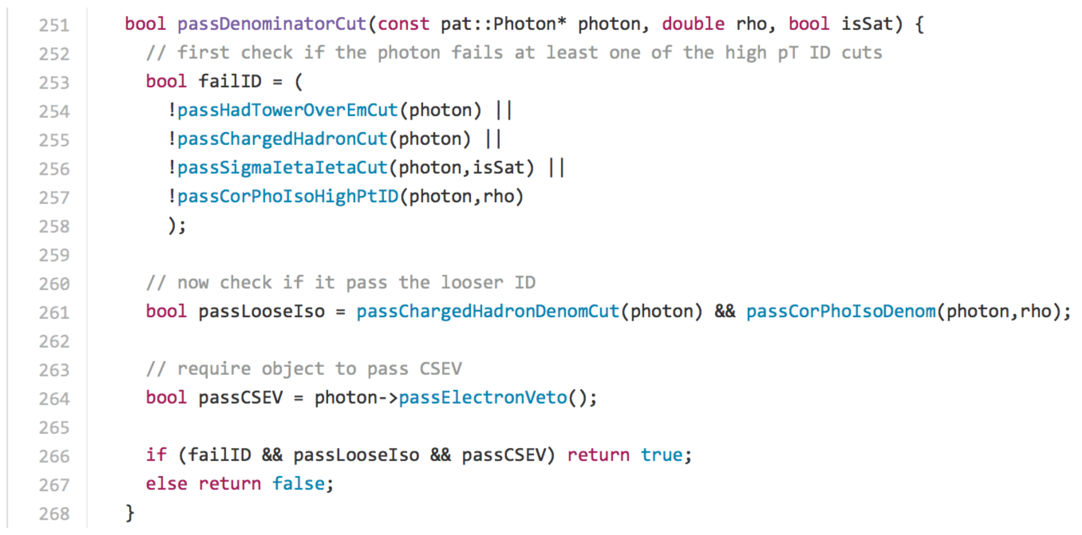
```
bool passDenominatorCut(const pat::Photon* photon, double rho, bool isSat) {
  // first check if the photon fails at least one of the high pT ID cuts
  bool failID = (
    !passHadTowerOverEmCut(photon) ||
    !passChargedHadronCut(photon) ||
    !passSigmaIetaIetaCut(photon,isSat) ||
    !passCorPhoIsoHighPtID(photon,rho)
    );

  // now check if it pass the looser ID
  bool passLooseIso = passChargedHadronDenomCut(photon) && passCorPhoIsoDenom(photon,rho);

  // require object to pass CSEV
  bool passCSEV = photon->passElectronVeto();

  if (failID && passLooseIso && passCSEV) return true;
  else return false;
}
```

The relaxed isolation cuts are



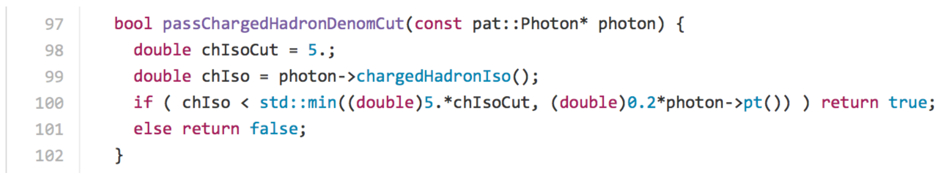
```
bool passDenominatorCut(const pat::Photon* photon, double rho, bool isSat) {
  // first check if the photon fails at least one of the high pT ID cuts
  bool failID = (
    !passHadTowerOverEmCut(photon) ||
    !passChargedHadronCut(photon) ||
    !passSigmaIetaIetaCut(photon,isSat) ||
    !passCorPhoIsoHighPtID(photon,rho)
    );

  // now check if it pass the looser ID
  bool passLooseIso = passChargedHadronDenomCut(photon) && passCorPhoIsoDenom(photon,rho);

  // require object to pass CSEV
  bool passCSEV = photon->passElectronVeto();

  if (failID && passLooseIso && passCSEV) return true;
  else return false;
}
```

and



```
bool passChargedHadronDenomCut(const pat::Photon* photon) {
  double chIsoCut = 5.;
  double chIso = photon->chargedHadronIso();
  if ( chIso < std::min((double)5.*chIsoCut, (double)0.2*photon->pt()) ) return true;
  else return false;
}
```

Real templates

Our real templates also come from selecting numerator objects

bool isNumeratorObj = Photon_isNumeratorObjCand && Photon_passChIso;

but are known real objects from MC.

Fake templates

Our fake templates come from numerator objects, but only those that fall into a certain side band of the charged hadron isolation variable. Offline we select

bool isFakeTemplateObj = Photon_isNumeratorObjCand && inChIsoSideband;

where

bool inChIsoSideband = (10. < Photon_chargedHadIso03) && (Photon_chargedHadIso03 < 15.);

Sigma_iEtaiEta

Since sigma_iEtaiEta isn't used in the numerator but is one of the ID-vetos in the denominator, it's possible for an object to end up in both our numerator and denominator definitions.

Our code

https://github.com/cms-exotica-diphotons/diphoton-analysis/blob/master/CommonCla

-- AndrewBuccilli - 2016-03-07

This topic: Main > NumAndDenomDefs
Topic revision: r4 - 2016-03-23 - AndrewBuccilli