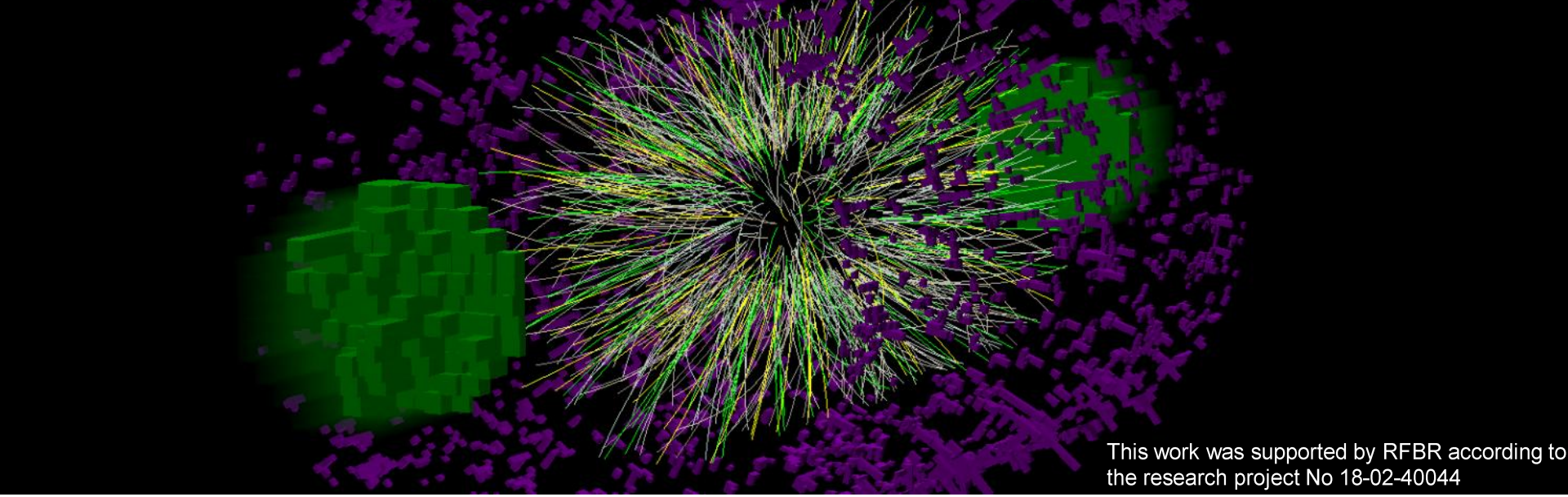


This work was supported by RFBR according to the research project No 18-02-40044

Looking at Data Stored in MpdDst

Pavel Batyuk[1], Olga Kodolova[2], Lyudmila Malinina[1,2], Konstantin Mikhaylov[1,3], Grigory Nigmatkulov[4]

[1]Joint Institute for Nuclear Research, Dubna, Russia,
[2]Skobeltsyn Research Institute of Nuclear Physics, Moscow State University, Moscow, Russia,
[3]NRC Kurchatov Institute – ITEP, Moscow, Russia
[4]National Research Nuclear University MEPhI, Moscow, Russia

November 21, 2019

Outline

Vertex reconstruction

Tracks:

- Global tracks (DCA to primary vertex)
- Primary tracks

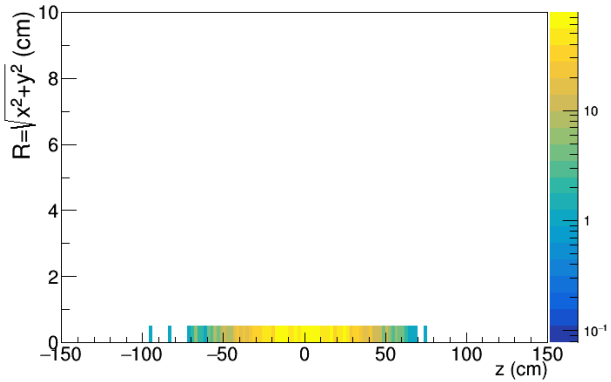
Summary/Requests

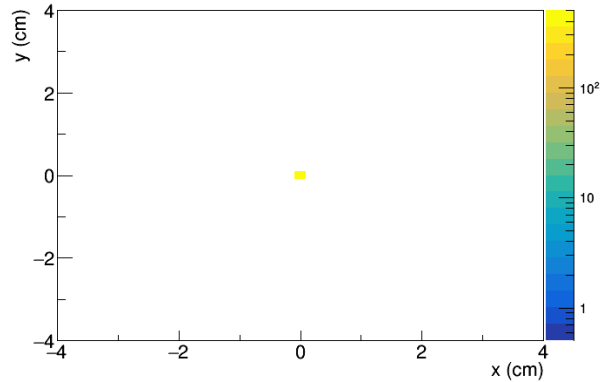
Data set (official production):
vHLLE+GEANT+reconstruction. Au+Au at $\sqrt{s_{NN}}$=7.7 GeV

Primary Vertex Reconstruction

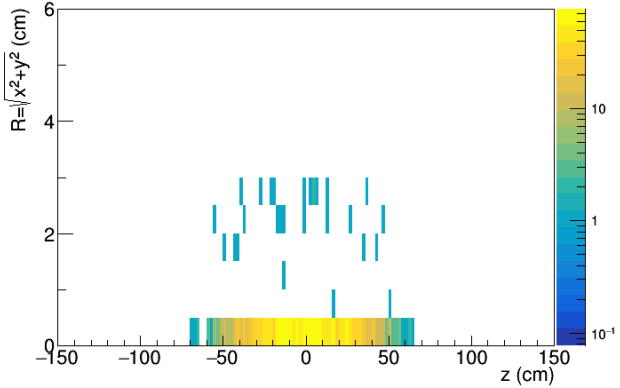
Primary Vertex Reconstruction

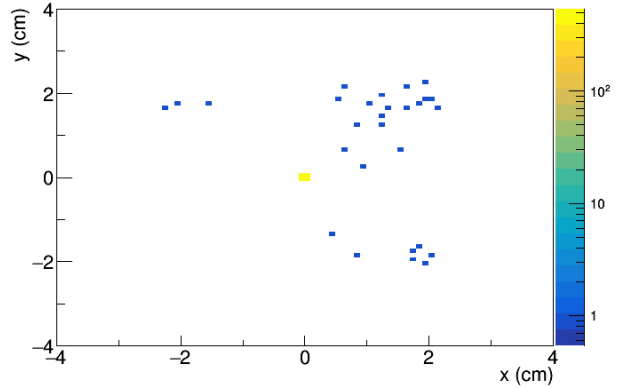
Simulated





Reconstructed





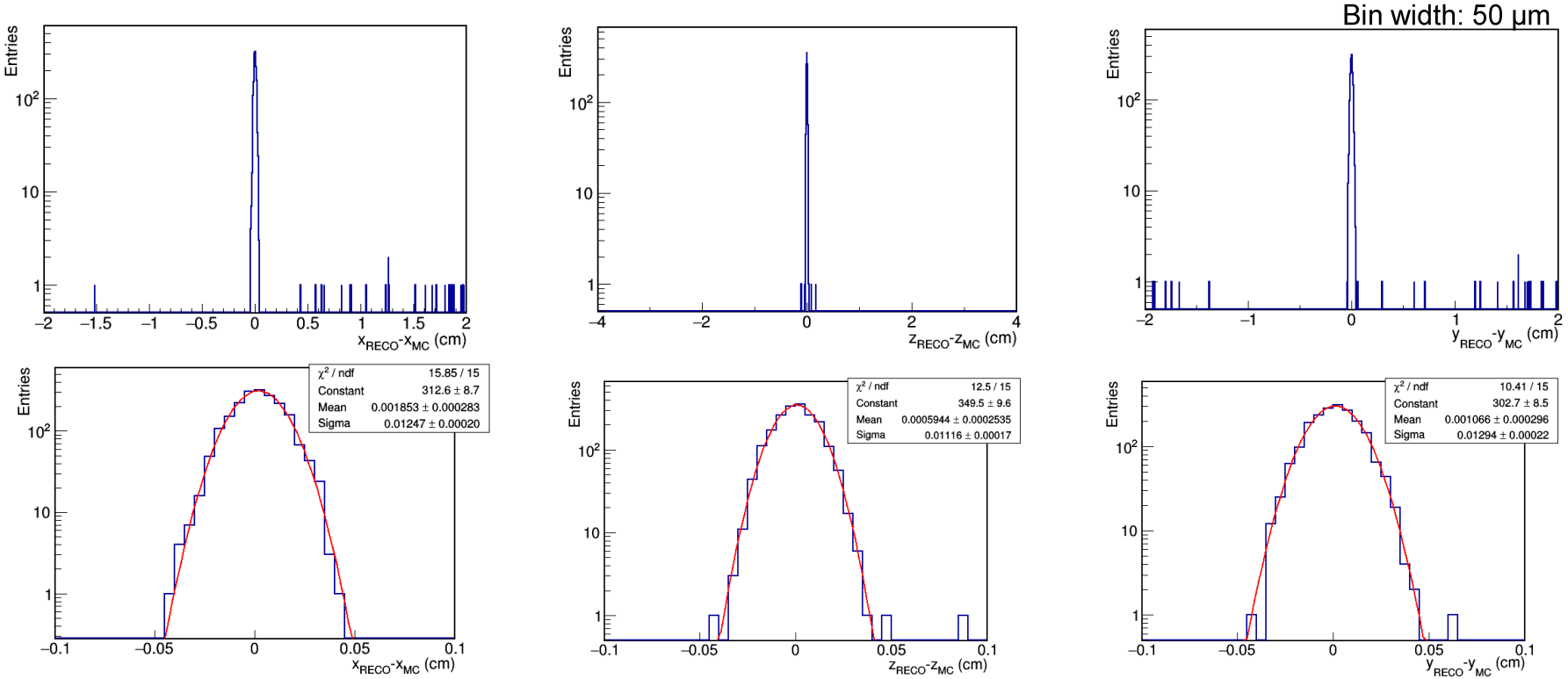
In the future it would be good to smear not only z-position of the primary vertex, but also x and y:

- More realistic scenario

Currently, not all reconstruction patterns are clear.

What is the size of the beam pipe?
What is the radial distance to the closest material? To the inner field cage?

Primary Vertex Reconstruction



Where are the outliers come from?

In general, looks reasonable for central collisions. Need tests for peripheral/p+p ones.

Tracks

```
//_____________________
bool isGoodTrack(int nHits, TVector3 mom, int charge) {
  return ( nHits >= 20 &&
           0.1 <= mom.Perp() && mom.Perp() <= 2.1 &&
           TMath::Abs( mom.Eta() ) <= 1. &&
           charge !=0 );
}
```

Track Types in MpdDst

There are several track types that are currently stored in MpdDst:

- Global
- Primary (container exists, but information is not stored)
- MC
- TPC Kalman

Global tracks are filled from Kalman ones. Amount is equal almost all the time…

How Kalman tracks were fitted? To beam line?

What information is stored in the Kalman track?

Naive expectation says that number of track for the classes should not differ much, BUT…

```
Track number [global/mc/kalman]:   [ 1015 / 54394 / 1015 ]
Track number [global/mc/kalman]:   [ 993 / 53730 / 993 ]
Track number [global/mc/kalman]:   [ 994 / 54140 / 994 ]
Track number [global/mc/kalman]:   [ 965 / 53991 / 965 ]
Track number [global/mc/kalman]:   [ 1074 / 60090 / 1074 ]
```

What is a definition of MCTrack?

What is Stored in DCA of Global Track?

MpdFillDstTask.cxx

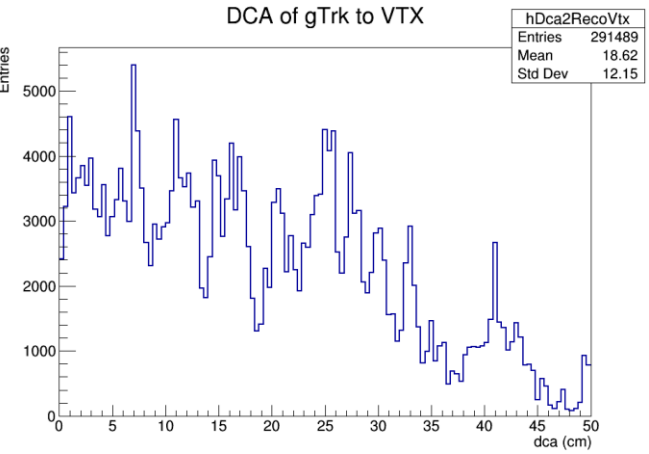
```
Double_t phi = kftrack->GetParam(0) / kftrack->GetPosNew();
track->SetFirstPointX(kftrack->GetPosNew() * TMath::Cos(phi)); // closest to beam line
track->SetFirstPointY(kftrack->GetPosNew() * TMath::Sin(phi));
track->SetFirstPointZ(kftrack->GetParam(1));

track->SetLastPointX(0.); // AZ - currently not available
track->SetLastPointY(0.); // AZ - currently not available
track->SetLastPointZ(0.); // AZ - currently not available
FillTrackDCA(track, &recoVertex, &mcVertex);
FillTrackPID(track);
FillTrackTpcHits(i, track);
```

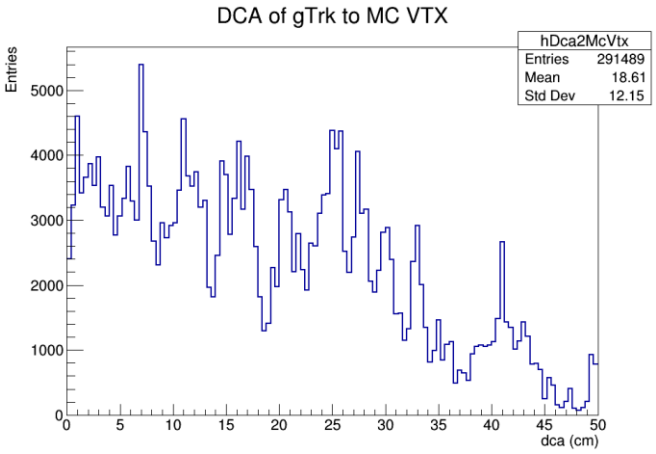
Side note:
In case one will want to use barrel electromagnetic calorimeter, the last point should be stored.

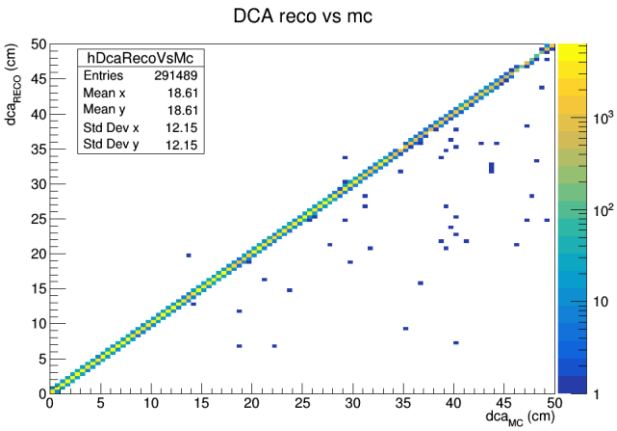
All projections should be done using track momentum estimated from this point (with material taken into account).

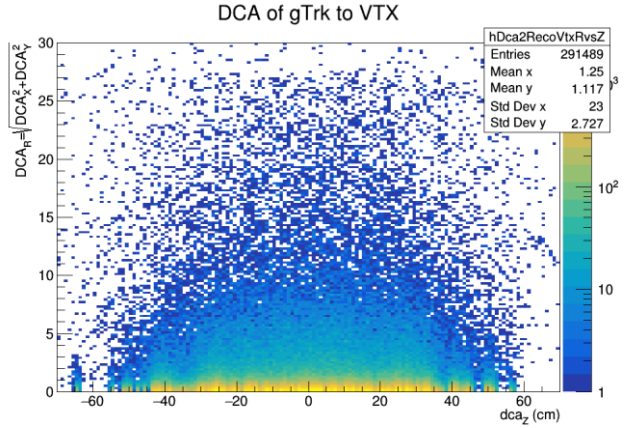
```
void MpdFillDstTask::FillTrackDCA(MpdTrack* track, TVector3 *recoVertex, TVector3 *mcVertex) {
    MpdHelix helix = track->GetHelix();
    Double_t path_at_mcVertex;
    Double_t path_at_recoVertex;
    path_at_mcVertex = helix.pathLength(*mcVertex);
    path_at_recoVertex = helix.pathLength(*recoVertex);
    TVector3 DCA_MC = helix.at(path_at_mcVertex);
    TVector3 DCA_RECO = helix.at(path_at_recoVertex);
    // set dca global as dca to MC vertex DW
    track->SetDCAGlobalX(DCA_MC.X()-mcVertex->X());
    track->SetDCAGlobalY(DCA_MC.Y()-mcVertex->Y());
    track->SetDCAGlobalZ(DCA_MC.Z()-mcVertex->Z());
    // set dca as dca to reconstructed vertex DW
    track->SetDCAX(DCA_RECO.X()-recoVertex->X());
    track->SetDCAY(DCA_RECO.Y()-recoVertex->Y());
    track->SetDCAZ(DCA_RECO.Z()-recoVertex->Z());
}
```

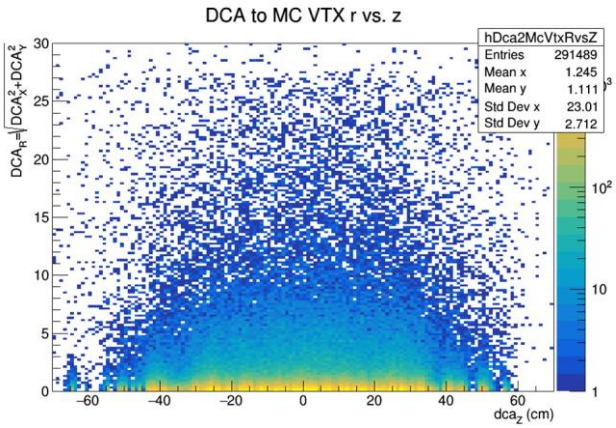
Lets Take a Closer Look



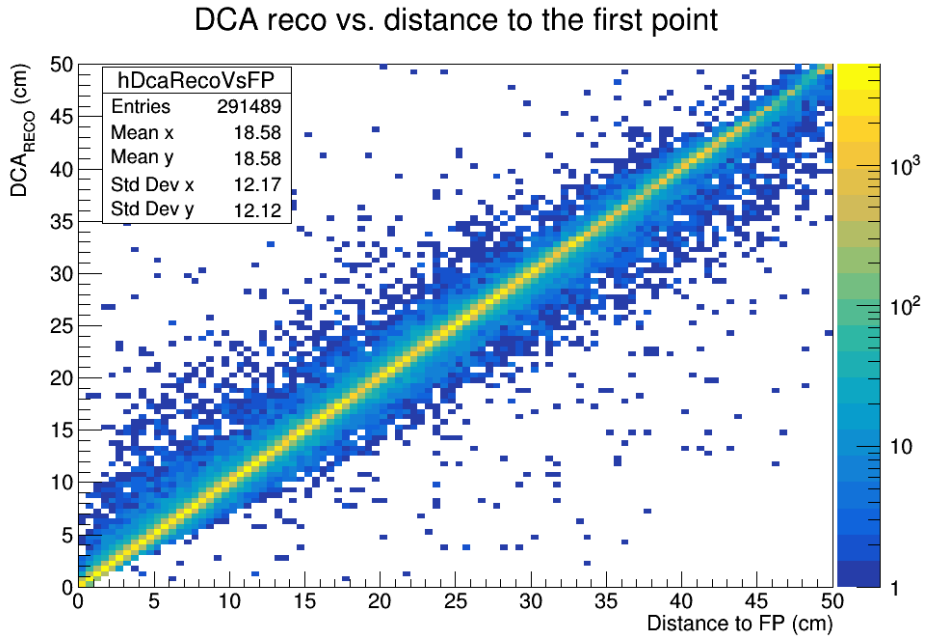


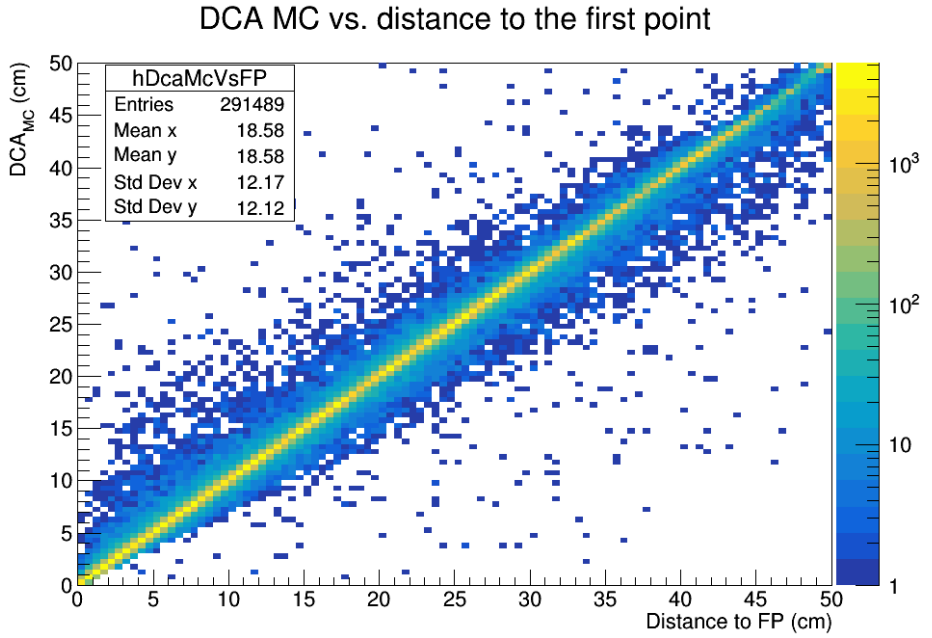






Lets Take a Closer Look





The idea is usual. Take the first point and momentum there (fitted point with the fitted momentum), make helix, find DCA of the helix to the primary vertex, take this point for the starting point (i.e. can estimate DCA) and look at the momentum of the track at this point. Momentum components change along the helix.

Helix Parameterization

Assumption: B-field = 0.5 T

The way it is implemented in the current version of MPD (one can use MpdHelix instead):

```
MpdMiniPhysicalHelix makeHelix(const MpdKalmanTrack *tr) {

  const Double_t F_CUR0 = 0.3 * 0.01 * 5 / 10; // 5kG
  double r = tr->GetPosNew();
  double phi = tr->GetParam(0) / r;
  double x = r * TMath::Cos(phi);
  double y = r * TMath::Sin(phi);
  double dip = tr->GetParam(3);
  double cur = F_CUR0 * TMath::Abs (tr->GetParam(4));
  TVector3 o(x, y, tr->GetParam(1));
  int h = (Int_t) TMath::Sign(1.1,tr->GetParam(4));
  MpdMiniPhysicalHelix helix(cur, dip, tr->GetParam(2)-TMath::PiOver2()*h, o, h);
  return helix;
}
```

```
MpdMiniPhysicalHelix gHelix = makeHelix(kalmanTrack);
double pathLength = gHelix.pathLength(recoVtxPos);
TVector3 gDca;
gDca = gHelix.at(pathLength);
gDca -= recoVtxPos;
```

The STAR/ALICE way of the calculation using MpdMiniPhysicalHelix (a part of MpdMiniDst):

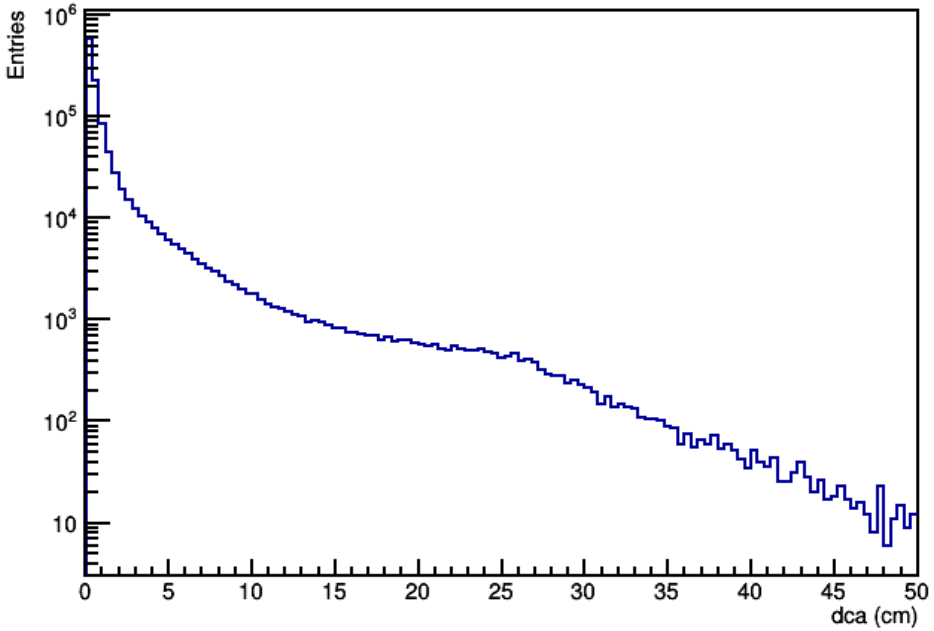
```
// Obtain helix from Kalman track
MpdMiniPhysicalHelix oHelix( mpdTrackMom, firstPoint, bField * kilogauss, charge);
double pathLength1 = oHelix.pathLength(recoVtxPos);
TVector3 oDca;
oDca = oHelix.at(pathLength1);
oDca -= recoVtxPos;
```

Both methods provide almost the same results with the accuracy of the rounding precision.

BUT...
In both cases, track momentum used for the estimation (either from Kalman or global track) is ALREADY stored for DCA point to the primary vertex.

Why? Why store components of the same track in different frames?

DCA of Global Track to Primary Vertex



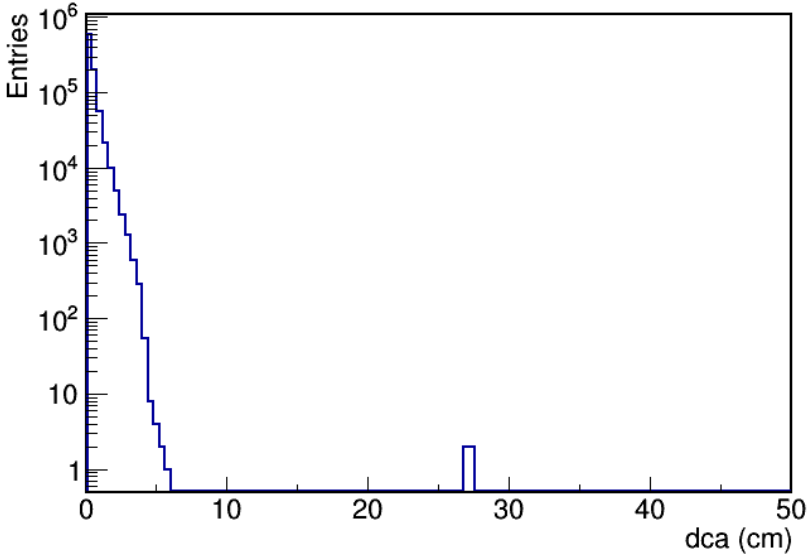
- Finally, a reasonable DCA distribution of global track to primary vertex! YEAH!!!
- But one still needs primary tracks to perform most of the analyses because their momentum distributions will differ from that for globals (momentum-dependent).
- Is there a way to retrieve them right now? … Yes (and no) ...

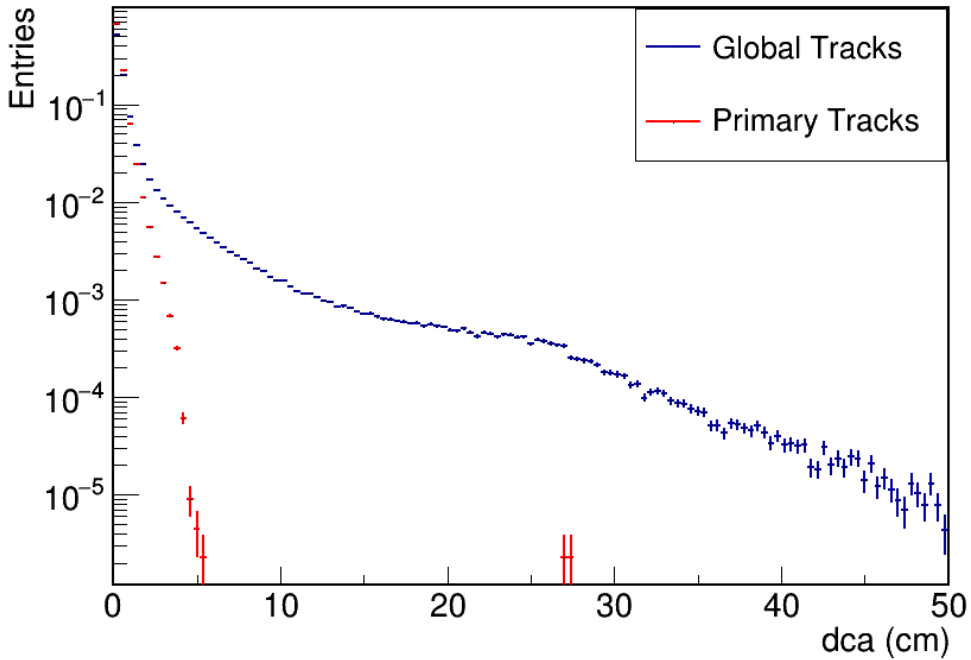
Looking at Tracks Used for the Vertex Reconstruction

```
// MpdVertex position (in case of several vertices)
MpdVertex *mpdVertex = (MpdVertex*)vertices->First();
TVector3 pVtx; mpdVertex->Position( pVtx );

// Return position of the reconstructed primary vertex
recoVtxPos.SetXYZ( pVtx.X(), pVtx.Y(), pVtx.Z() );

// Retrieve number of kalman tracks that were used for the vertex fit
int nVtxTracks = mpdVertex->GetNTracks();

// Loop over pointers to Kalman tracks
for ( int iTrk=0; iTrk<nVtxTracks; iTrk++ ) {

  //std::cout << "Kalman vtx track index: " <<  mpdVertex->GetIndices()->At(iTrk) << std::endl;

  // Retrieve Kalman track
  MpdKalmanTrack *kalmanTrack = (MpdKalmanTrack*)kalmanTracks->UncheckedAt( mpdVertex->GetIndices()->At(iTrk) );

  // Check that track exists
  if ( !kalmanTrack ) continue;

  // Check good track (?)
  if ( !isGoodTrack( kalmanTrack->GetNofHits(), kalmanTrack->Momentum3(), kalmanTrack->Charge() ) ) continue;

  MpdMiniPhysicalHelix gHelix = makeHelix(kalmanTrack);
  double pathLength = gHelix.pathLength(recoVtxPos);
  TVector3 gDca;
  gDca = gHelix.at(pathLength);
  gDca -= recoVtxPos;
  // DCA to VTX from MPD helix
  hMpdPrimDcaMag->Fill( gDca.Mag() );
} // for ( int iTrk=0; iTrk<nVtxTracks; iTrk++ )
```

```
Track number [global/mc/kalman/vtxtrack]:   [ 1124 / 59001 / 1124 / 804]
Track number [global/mc/kalman/vtxtrack]:   [ 279 / 58446 / 279 / 22]
Track number [global/mc/kalman/vtxtrack]:   [ 1179 / 62250 / 1179 / 867]
Track number [global/mc/kalman/vtxtrack]:   [ 154 / 60929 / 154 / 13]
Track number [global/mc/kalman/vtxtrack]:   [ 1148 / 60905 / 1148 / 842]
Track number [global/mc/kalman/vtxtrack]:   [ 1012 / 58294 / 1012 / 743]
Track number [global/mc/kalman/vtxtrack]:   [ 1124 / 59140 / 1124 / 827]
Track number [global/mc/kalman/vtxtrack]:   [ 1053 / 56285 / 1053 / 795]
Track number [global/mc/kalman/vtxtrack]:   [ 1052 / 57745 / 1052 / 757]
```

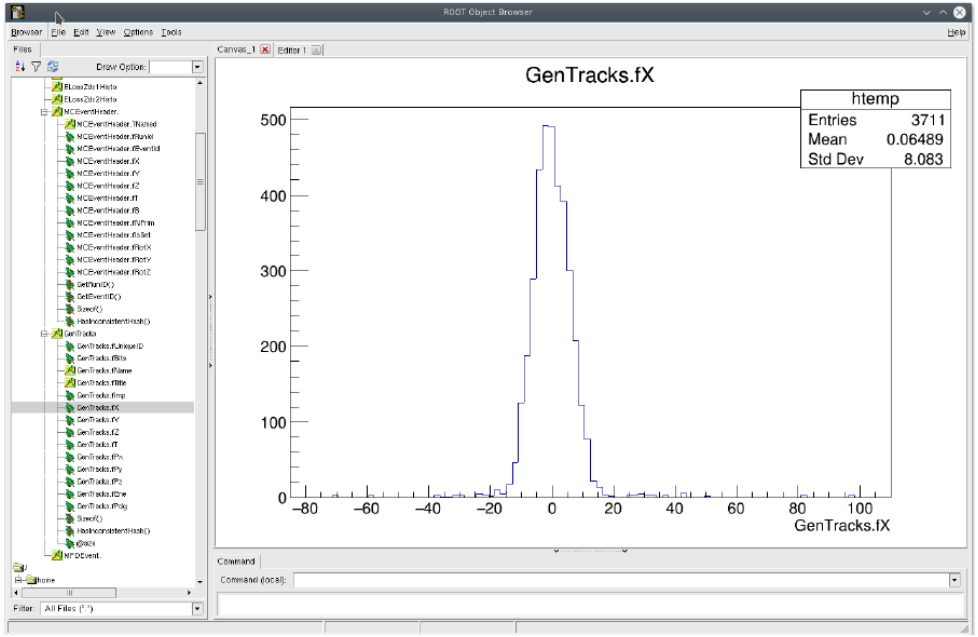
Primary Tracks





- Reasonable distribution except outliers. What is the origin of outliers?
- Need to store momentum of the primary tracks (global/kalman tracks FITTED to primary vertex)

Using information from generator in MpdDst



- McDst format is used as an unification of outputs from different generators (vHLLE+UrQMD and UrQMD at the moment)
- All necessary instruments to be used when doing detector simulations with the McDst format are ready (**generators/MpdMcDstGenerator.h**).

runMC.C:

```
#define MCDST

...

#ifdef MCDST // McDst generator
if (!CheckFileExist(inFile)) return;

MpdMcDstGenerator* mcDstGen =
new MpdMcDstGenerator(inFile);
primGen->AddGenerator(mcDstGen)
…
```

GenTracks branch (available with the McDst format as input):
(**mpdbase/MpdGenTrack.h, mpdbase/MpdGenTrackTask.h**)

- **Contains info (space-time, momentum ...) to be used mainly for femtoscopy.**
- **Written to the output DST.**

Summary/Requests

- Primary vertex reconstruction
  - Seems to be reasonable for central A+A collisions.
  - Need to understand outliers.
  - Need tests with peripheral/p+p collisions.
- Tracks
  - DCA information can be retrieved.
  - Primary tracks can be partially restored. Outliers have to be explained.
- Requests:
  - Code documentation: what is stored and at which conditions
    KalmanTrack, MCTrack, etc.
  - Information from trigger detectors: what is the T0? Which detectors provide
    start time for TOF? How the triggers will be stored? Need realistic simulation.
  - Primary track (fitted to primary vertex) information MUST be stored.
  - Need Monte Carlo information from generator level. Otherwise, there is
    no way of realistic efficiency/resolution estimation. Also important for physics analyses.