Tracking validation

Lia Lavezzi

2021-04-01

Tracking Quality Performances

A proposal of things to check in order to understand tracking quality

Stefano Spataro
11/01/2019

Cut&Paste from old presentations about tracking quality
(thanks to Lia Lavezzi@Panda and Bianca Scavino@BelleII)

DISCLAIMER: both PANDA and BelleII plots are obsolete, they are shown just as examples

- I would like to address the task which was introduced some time ago by Stefano
- I will briefly re-propose the study I would like to make, borrowing from his slides

Tracking validation

TRACKING = PATTERN RECOGNITION + TRACK FITTING

Associate the hits together in a **track candidate**

requirements:

- associate all the hits created by the same particle to the track candidate
- do not associate hits created by noise, background or another particles to the track candidate
- do not create clone tracks, i.e. duplicated of true tracks
- do not create ghost tracks, i.e. associate hits in a non existing tracks

In summary

Associate all and only the hits coming from a real particle to one correct track candidate

Fit the track candidate hits and obtain the **track** parametrization

requirements of prefit (helix)

- position and momentum at starting point/vertex, with a reasonable covariance matrix
- sort the hits along the track

requirements of Kalman fit

- track description with better resolution → five parameters → momentum, position and covariance matrix
- correct pull distribution
- cleanup track (DAF)

In summary

Find the best track hypothesis starting from the hit list

Pattern recognition

I would like to study the tracks coming from Hough transform, before the Kalman fit

SINGLE TRACK
→ **Defined on hits**

$$\text{EFFICIENCY} = \frac{\#\ \text{CORRECTLY ASSIGNED HITS}}{\#\ \text{MC POINTS}}$$

$$\text{PURITY} = \frac{\#\ \text{CORRECTLY ASSIGNED HITS}}{\#\ \text{RECO HITS}}$$

@Panda

SINGLE EVENT
→ **Defined on tracks**

$$\text{EFFICIENCY} = \frac{\#\ \text{TRACKS WITH SINGLE TRACK EFF} > 80\%}{\#\ \text{MC RECONSTRUCTABLE}^{*}\ \text{TRACKS}}$$

**reconstrubctable* means:
- 3 hits for xy
- 2 hits for $z\phi$

A **connection** between the hit used in PR and the MC point from which it was generated is necessary to:
- check if it is correctly assigned to the track
- check if it comes from noise
- check if it comes from background (when bk will be added)

→ it is necessary to evaluate efficiency/purity of the found track candidate

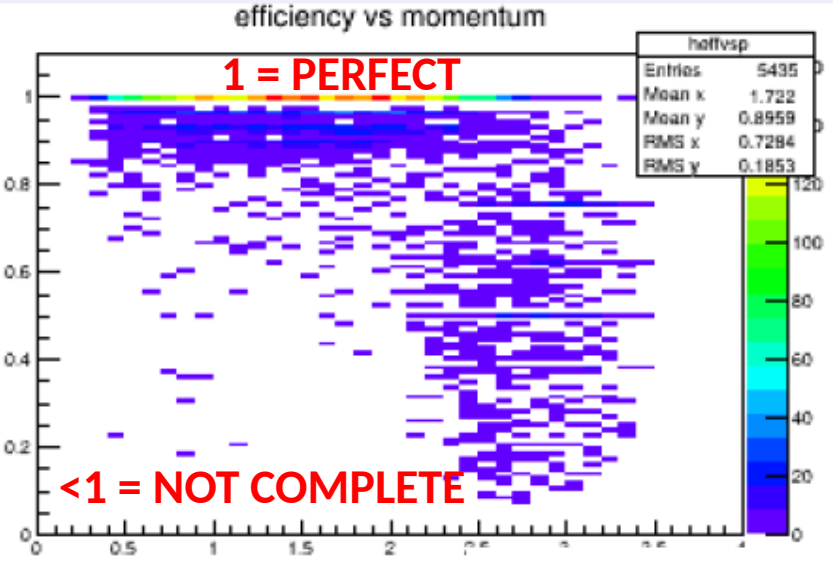
A match between the reconstructed and MC track is also important to evaluate the number of **reconstructable** tracks, i.e. tracks which leave a minimum number of MC points in the trackers

Pattern recognition

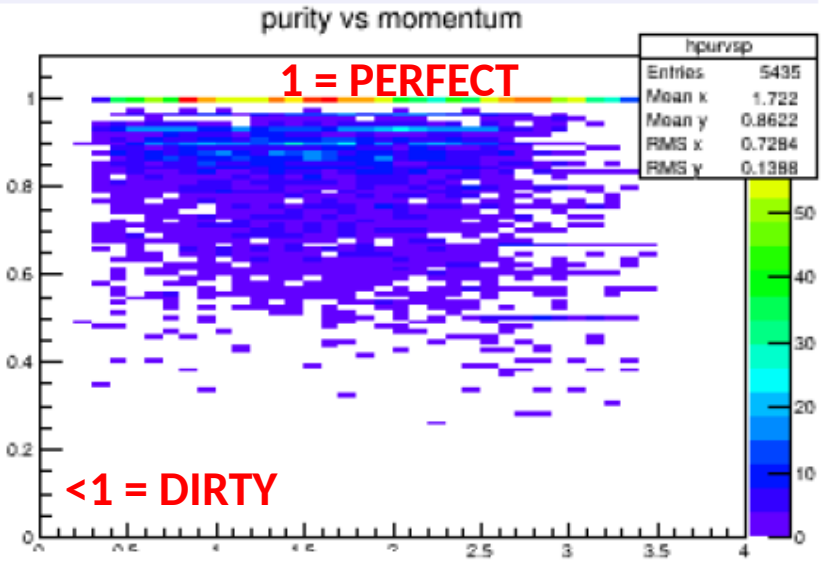
I would like to study the tracks coming from Hough transform, before the Kalman fit:

- efficienct/purity vs total momentum, transverse momentum, theta angle

$$\text{EFFICIENCY} = \frac{\#\ \text{CORRECTLY ASSIGNED HITS}}{\#\ \text{MC POINTS}}$$



$$\text{PURITY} = \frac{\#\ \text{CORRECTLY ASSIGNED HITS}}{\#\ \text{RECO HITS}}$$



Panda simulation of ΛΛ events

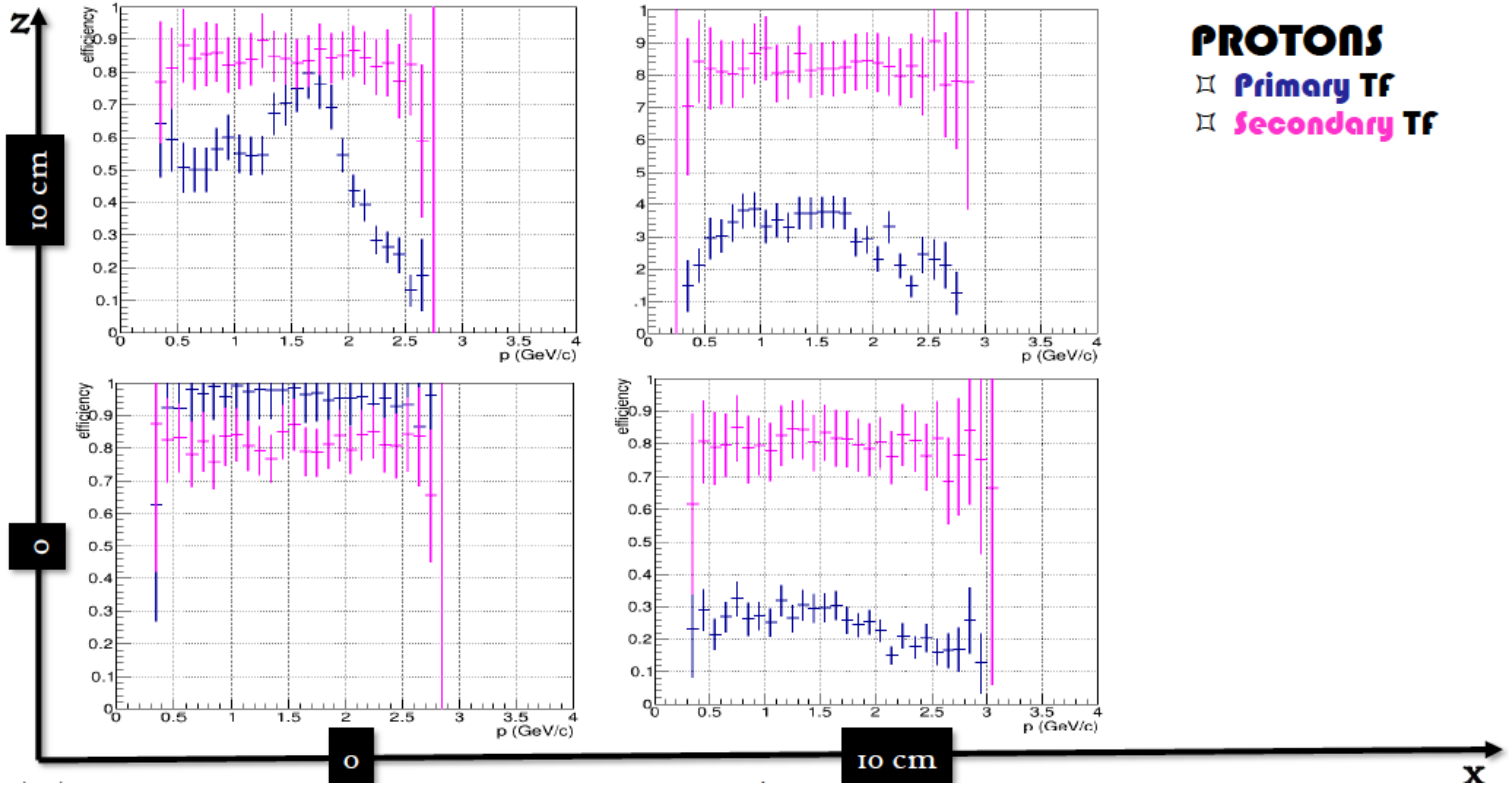
Plots like these highlight the problems, if there is any, and help in understanding the process step-by-step

Pattern recognition

4 pivotal layers

Efficiency vs momentum

$$\text{EFFICIENCY} = \frac{\#\ \text{TRUE TRACKS WITH} > 80\%\ \text{CORRECTLY ASSIGNED HITS}}{\#\ \text{MC RECONSTRUCTABLE TRACKS}}$$



- These plots, at track level, provide information on the uniformity of the efficiency w.r.t. track momentum and direction
- I would like to compile them with the tracks before Kalman fit and with different multiplicities (and maybe different particle types)

Status

I managed to write an algorithm class and retrieve the collections I need

```
// hough tracks before kalman
SmartDataPtr<RecMdcTrackCol> hough_track_Col(eventSvc(), "/Event/Recon/RecMdcTrackCol");

// kalman tracks
SmartDataPtr<RecMdcKalTrackCol> kalman_track_Col(eventSvc(),"/Event/Recon/RecMdcKalTrackCol");

// MDC digi
SmartDataPtr<MdcDigiCol> mdc_digi_Col(eventSvc(),"/Event/Digi/MdcDigiCol");

// MDC hit (come from digi)
SmartDataPtr<RecMdcHitCol> mdc_hit_Col(eventSvc(),"/Event/Recon/RecMdcHitCol");

// CGEM hit (i.e. cluster from Toy clusterizer)
SmartDataPtr<RecCgemClusterCol> cgem_cluster_Col(eventSvc(), "/Event/Recon/RecCgemClusterCol");

// cgem MC point
SmartDataPtr<Event::CgemMcHitCol> cgem_MC_point_Col(eventSvc(), "/Event/MC/CgemMcHitCol");

// mdc MC point
SmartDataPtr<Event::MdcMcHitCol> mdc_MC_point_Col(eventSvc(),"/Event/MC/MdcMcHitCol");

// MC particle
SmartDataPtr<Event::McParticleCol> MC_particle_Col(eventSvc(),"/Event/MC/McParticleCol");
```

Status

I iterate on the Hough track collection and get **RecMdcTrack**

```
// HOUGH TRACK
RecMdcTrackCol::iterator iter_hough_track = hough_track_Col->begin();
for(iter_hough_track = hough_track_Col->begin(); iter_hough_track != hough_track_Col->end(); iter_hough_track++) {
  RecMdcTrack *hough_track = (RecMdcTrack*) (*iter_hough_track);
```

I obtain from the RecMdcTrack the vectors of **RecMdcHit** and **RecCgemCluster**

```
//
// hit vector ----------------------------------
HitRefVec hitvector = hough_track->getVecHits();
ClusterRefVec clustervector = hough_track->getVecClusters();
```

I would like to retrieve from the **RecMdcHit** and **RecCgemCluster** the information on the corresponding **MdcMCHit** and **CgemMCHit**