Implementation of CKF for CDC and SVD TF

Sasha Glazov, Miraim Kuenzel, Aiqiang Guo

F2F tracking meeting, KEK, 2018 Feb

Work flow of CKF based TF

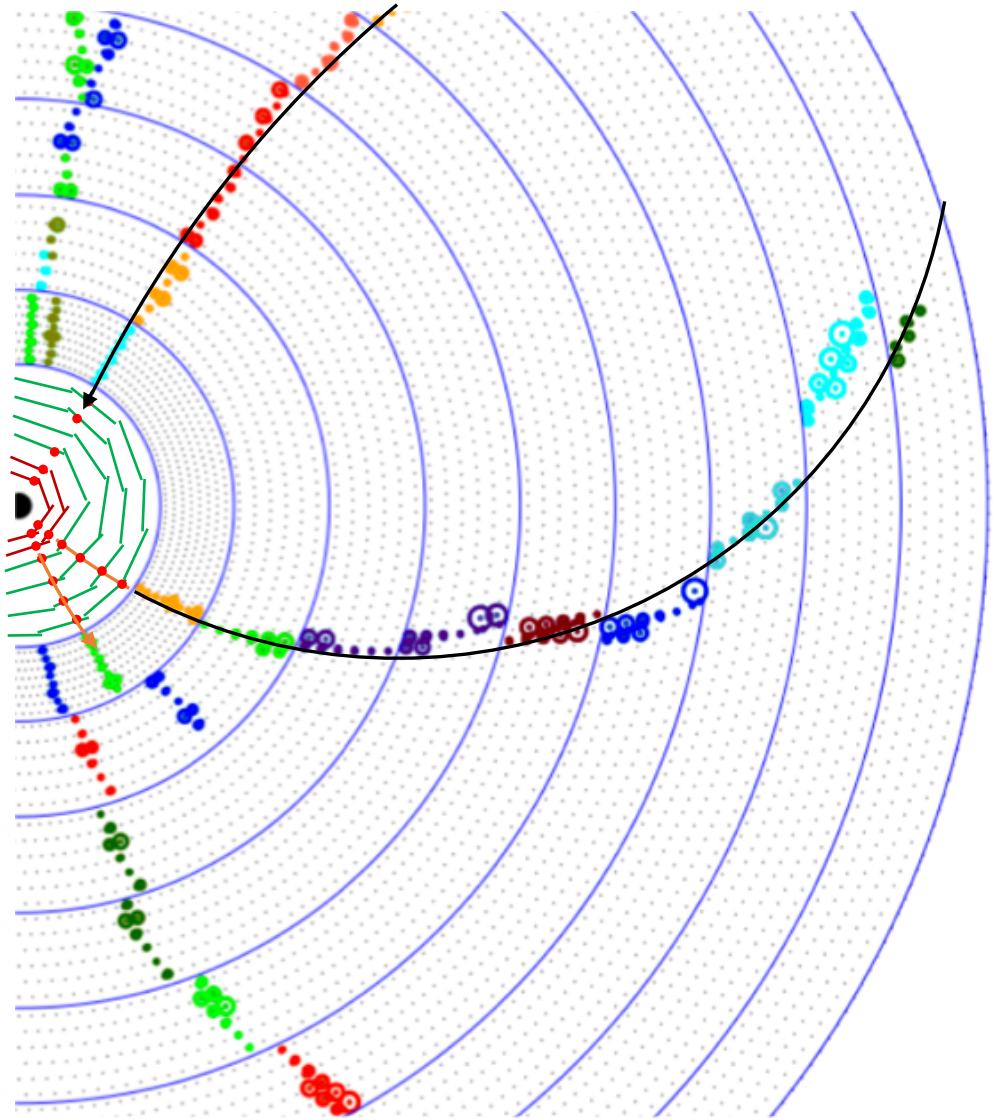
Implemented by Nils

1. Stand-alone CDC tracking
2. Stand-alone SVD tracking
3. Merge CDC →SVD
4. For unmerged CDC track, add extra SVD hits with CKF

5. For unmerged SVD track, add extra CDC hits with CKF

Miraim is working on it

6. Extrapolate to PXD with CKF



CKF based tracking validation

- MC: Y(4S) events

b2run_simulation -p 10 -d TrackingDetector -n 1000 -g generic generic1000.root

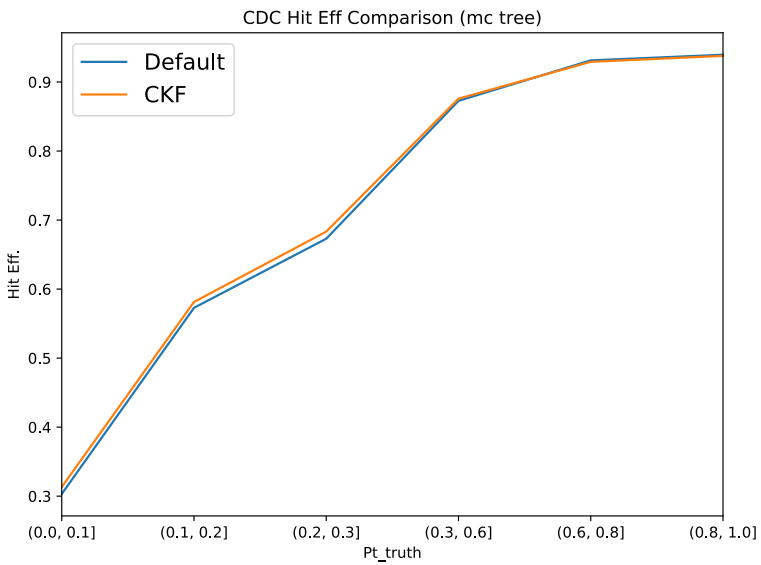
Default track finding

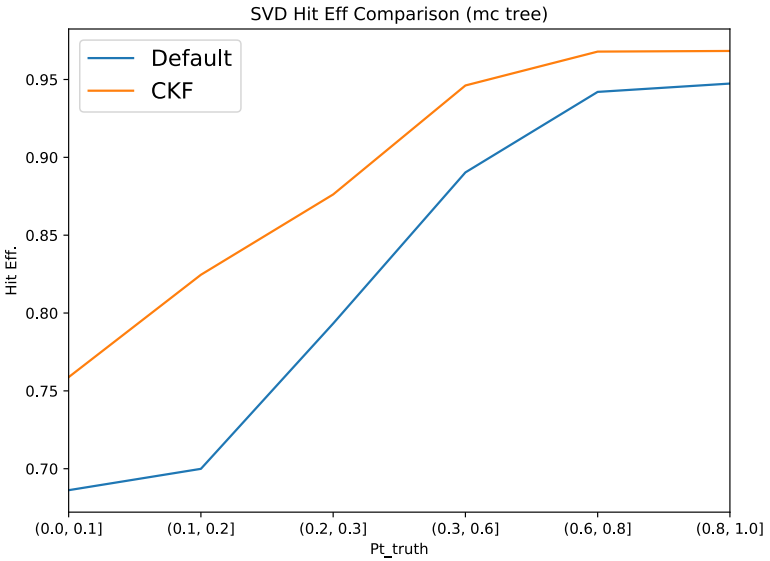
```
import basf2
from tracking import add_track_finding, add_mc_matcher
from tracking.harvesting_validation.combined_module import
CombinedTrackingValidationModule

from svd import add_svd_reconstruction
from pxd import add_pxd_reconstruction

path = basf2.create_path()

path.add_module("RootInput", inputFileName="generic1000.root")

path.add_module("Gearbox")
path.add_module("Geometry", useDB=True)
path.add_module("SetupGenfitExtrapolation")

add_svd_reconstruction(path)
add_pxd_reconstruction(path)

add_track_finding(path)
add_mc_matcher(path)

path.add_module(CombinedTrackingValidationModule("", "",
"output_validation.root", expert_level=200))

basf2.process(path)
```

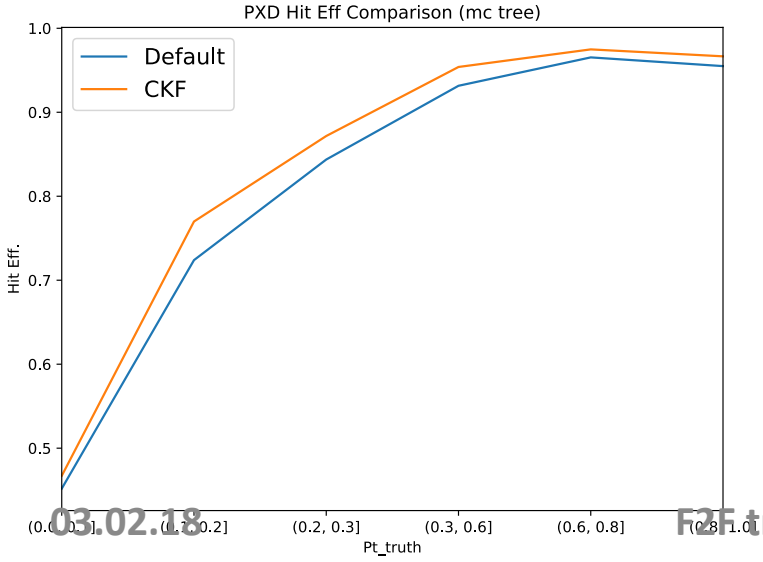
CKF based track finding

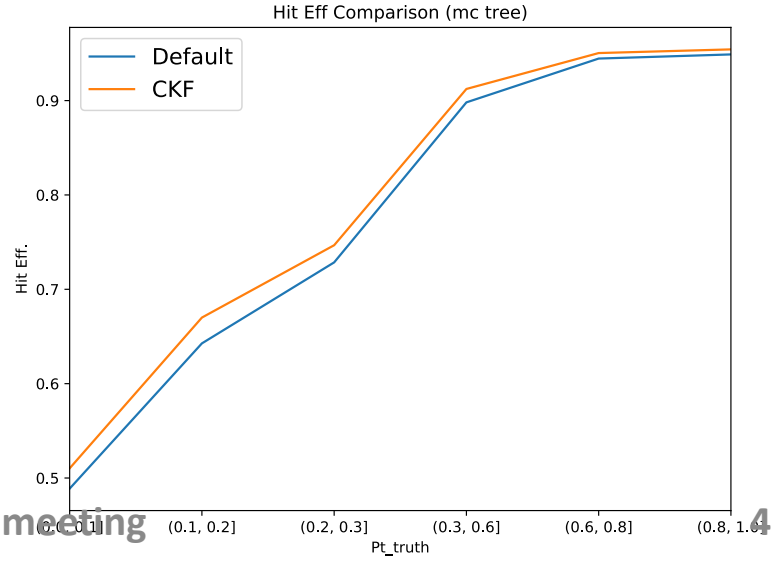
```
import basf2
from tracking import add_track_finding, add_mc_matcher,
add_ckf_based_track_finding
from tracking.harvesting_validation.combined_module import
CombinedTrackingValidationModule

from svd import add_svd_reconstruction
from pxd import add_pxd_reconstruction

path = basf2.create_path()

path.add_module("RootInput", inputFileName="generic1000.root")

path.add_module("Gearbox")
path.add_module("Geometry", useDB=True)
path.add_module("SetupGenfitExtrapolation")

add_svd_reconstruction(path)
add_pxd_reconstruction(path)

add_ckf_based_track_finding(path)
add_mc_matcher(path)

path.add_module(CombinedTrackingValidationModule("", "",
"output_validation_ckf.root", expert_level=200))

basf2.process(path)
```

Hit Efficiency

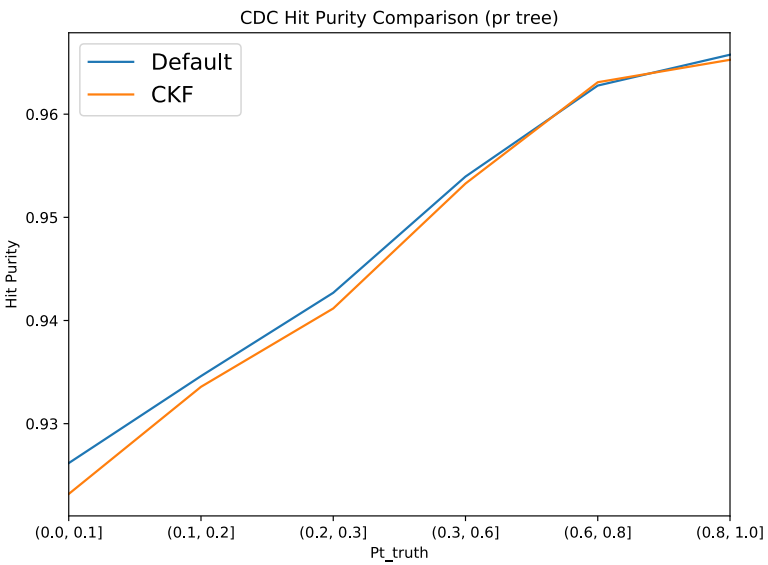


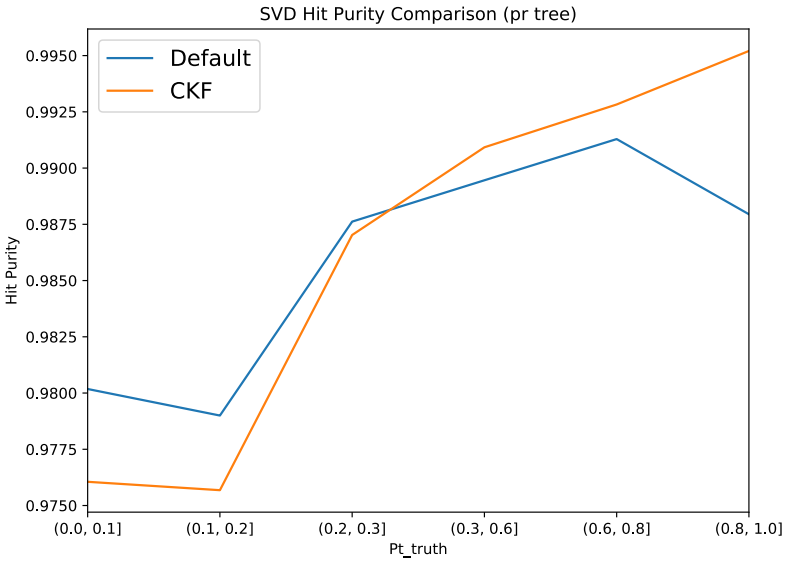


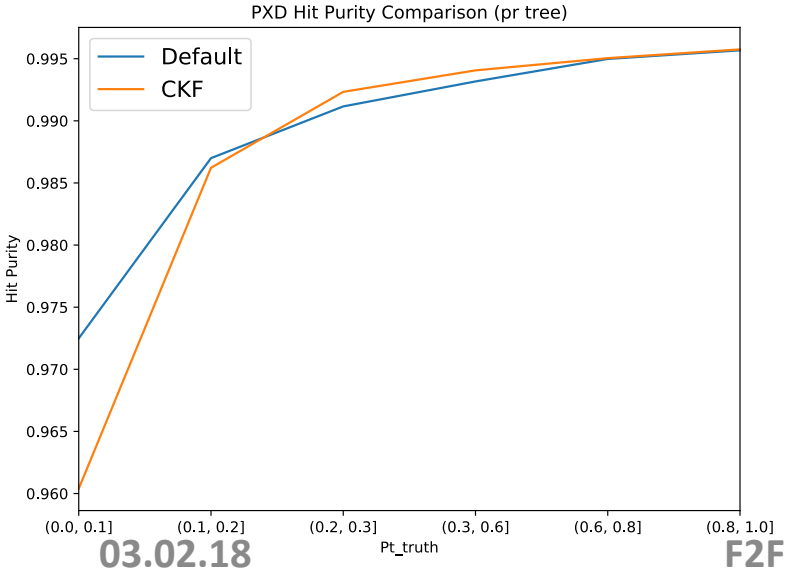


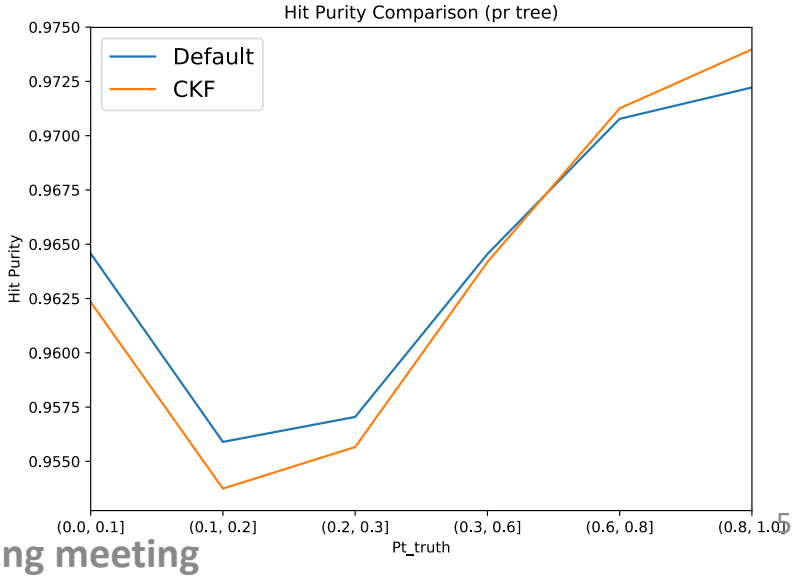


Hit purity

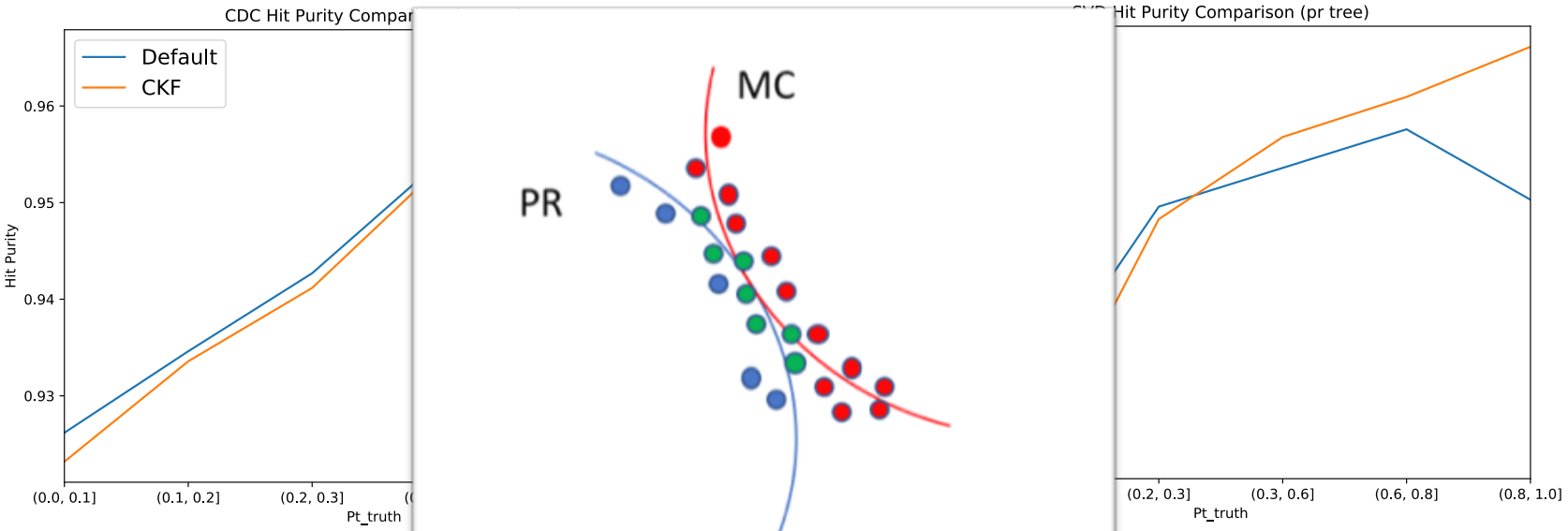

CDC Hit Purity Comparison (pr tree)
Default
CKF
Hit Purity
0.96
0.95
0.94
0.93
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth


SVD Hit Purity Comparison (pr tree)
Default
CKF
Hit Purity
0.9950
0.9925
0.9900
0.9875
0.9850
0.9825
0.9800
0.9775
0.9750
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth


PXD Hit Purity Comparison (pr tree)
Default
CKF
Hit Purity
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.960
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth

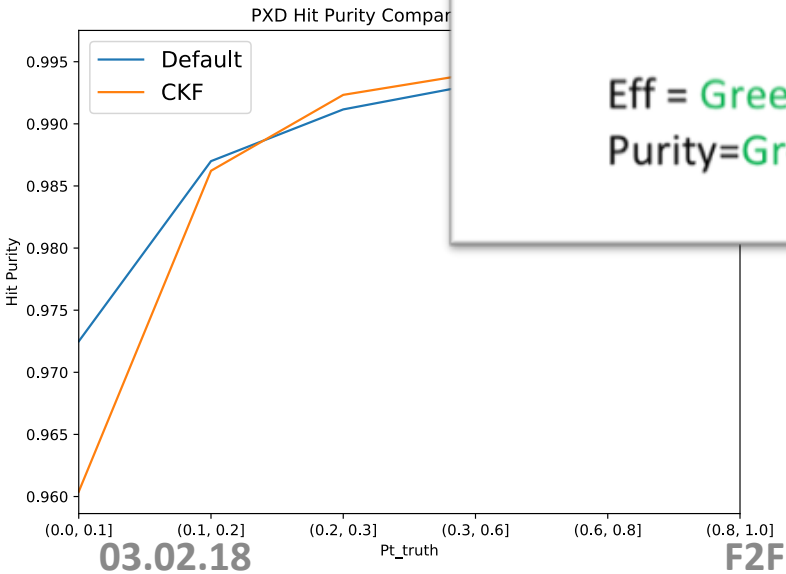

Hit Purity Comparison (pr tree)
Default
CKF
Hit Purity
0.9750
0.9725
0.9700
0.9675
0.9650
0.9625
0.9600
0.9575
0.9550
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth

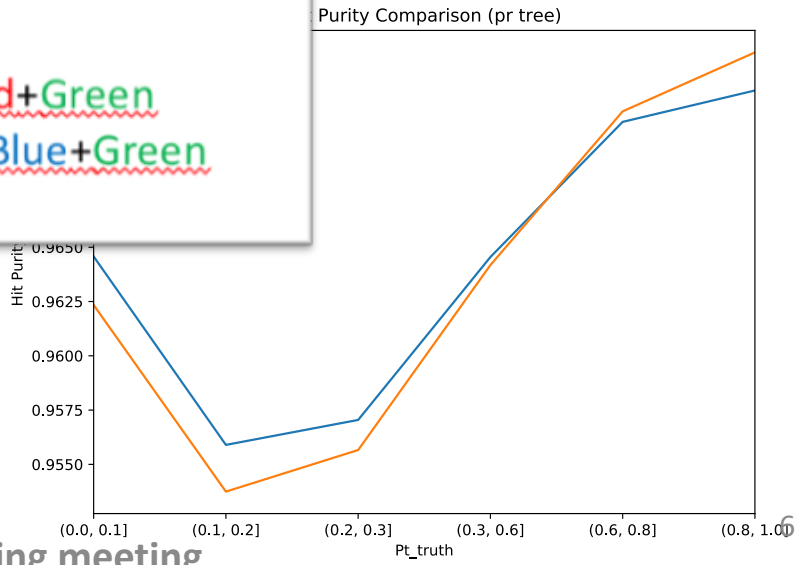
Hit purity


CDC Hit Purity Compar
Default
CKF
Hit Purity
0.96
0.95
0.94
0.93
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
Pt_truth

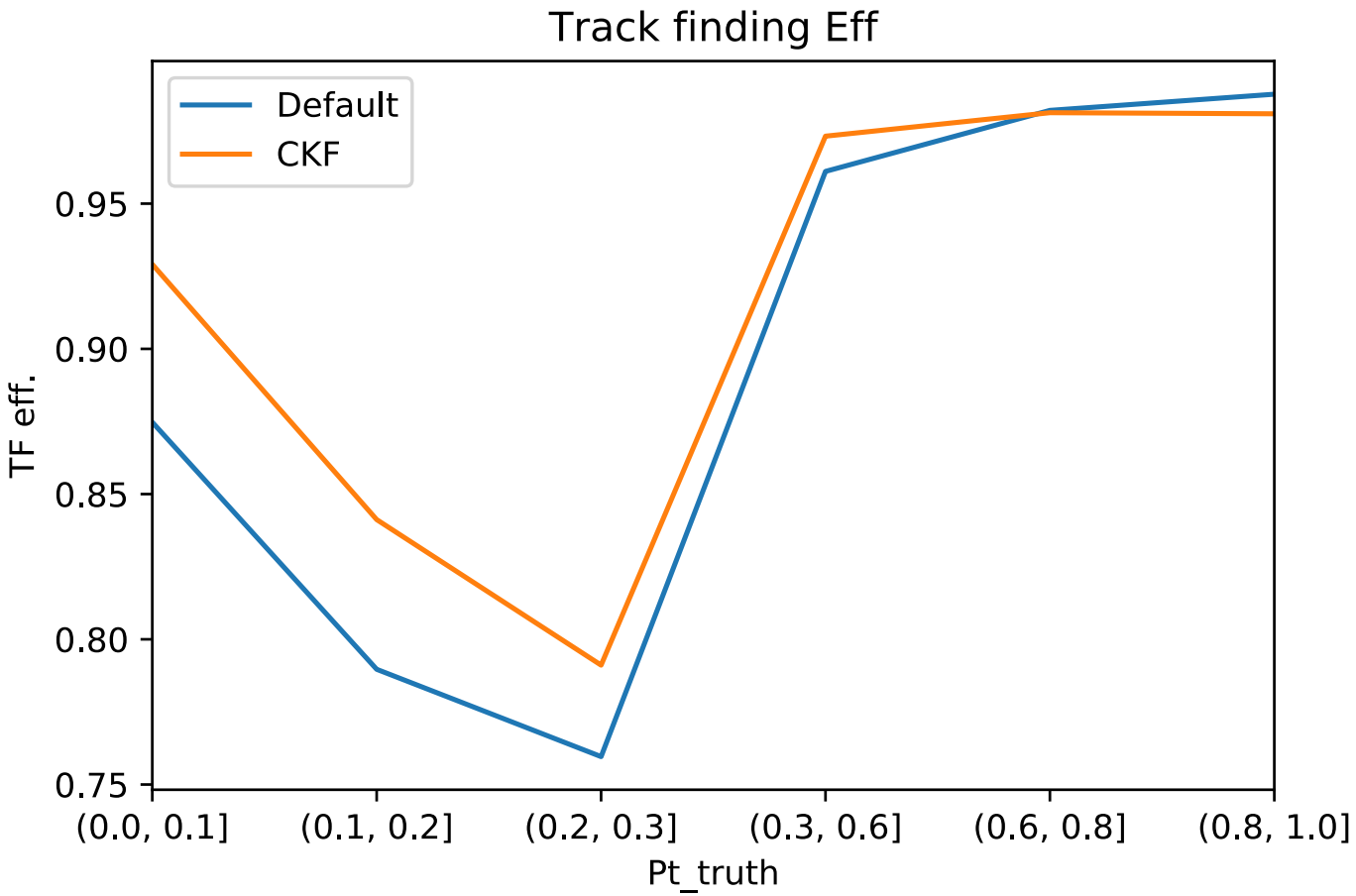
MC
PR
Eff = Green/Red+Green
Purity=Green/Blue+Green

Hit Purity Comparison (pr tree)
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth


PXD Hit Purity Compar
Default
CKF
Hit Purity
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.960
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth

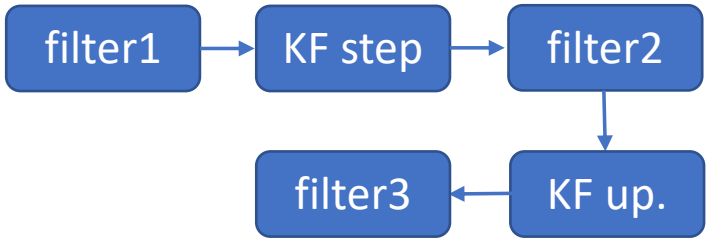

Purity Comparison (pr tree)
0.9650
0.9625
0.9600
0.9575
0.9550
(0.0, 0.1]
(0.1, 0.2]
(0.2, 0.3]
(0.3, 0.6]
(0.6, 0.8]
(0.8, 1.0]
Pt_truth

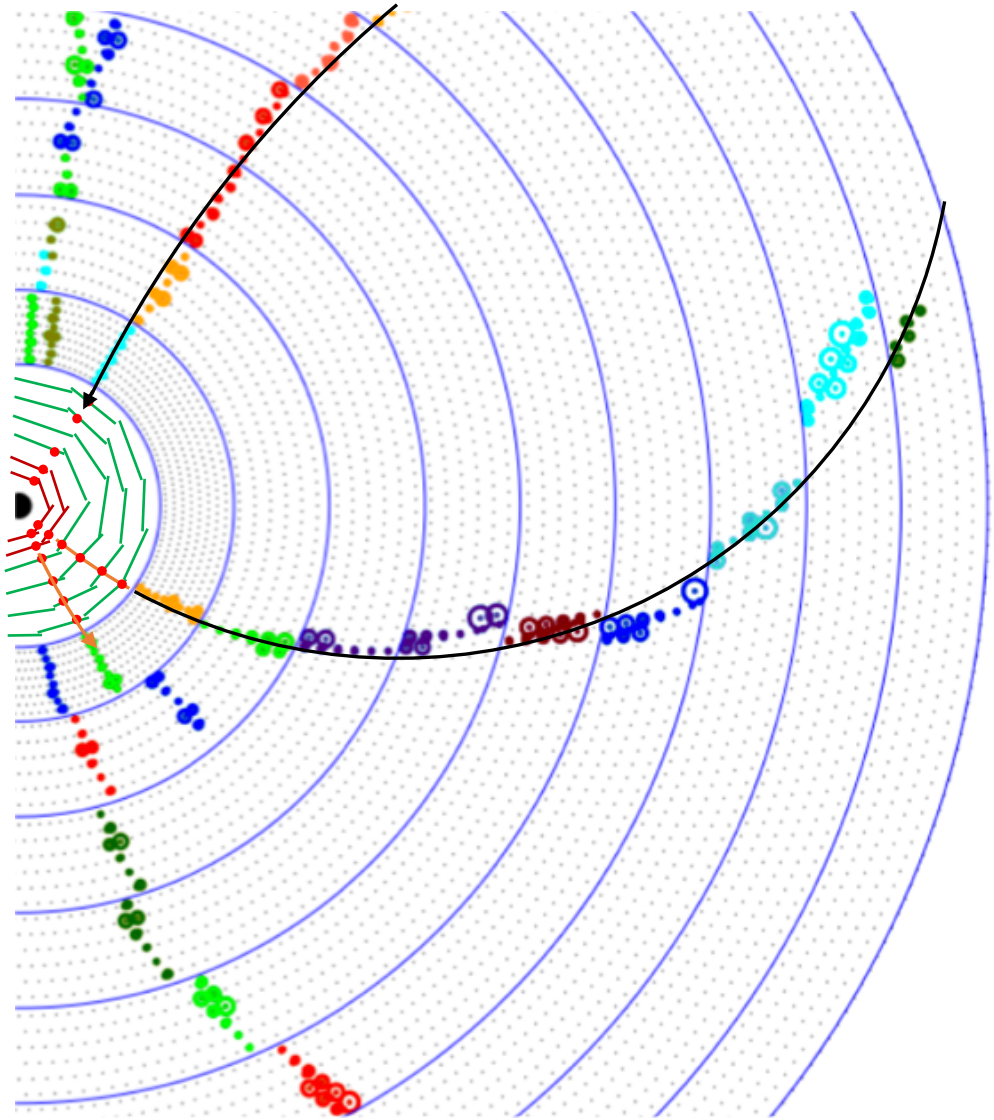
TF efficiency



SVD to CDC CKF

1. Stand-alone CDC tracking
2. Stand-alone SVD tracking
3. Merge CDC →SVD (CKF)
4. For unmerged CDC track, add extra SVD hits with CKF
5. For unmerged SVD track, add extra CDC hits with CKF
6. Extrapolate to PXD with CKF





The implementation

The current strategy is as follows:

- Load all the needed items: VXD seed tracks and CDC hits
- Turn every loaded object into one CKF state
- Implement the seed-hit and hit-hit relations(those may be based on clusters, segments or layers)
- Hits allowed by the relation maps have to survive the following chain: filter 1 - Kalman step - filter 2 -Kalman update - filter 3(Use NResults where possible)
- The result is written out as one (preferred) or more reco tracks

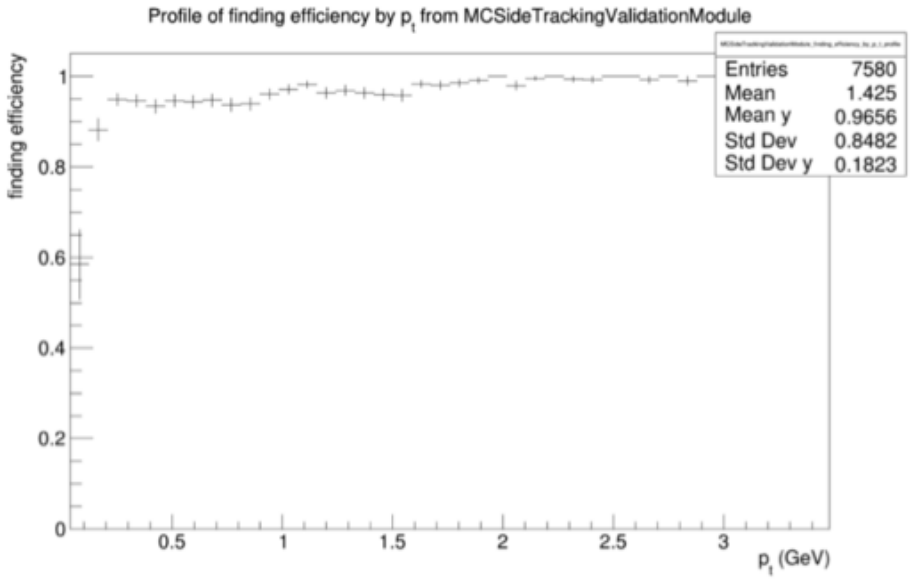
The filter variables

- Filter 1 and 2 (already fully implemented) will be distance-based filters, filter 3 will probably be a χ^2 filter
  - Filter 1: distance xy
  - Filter 2: distance mSoP xy

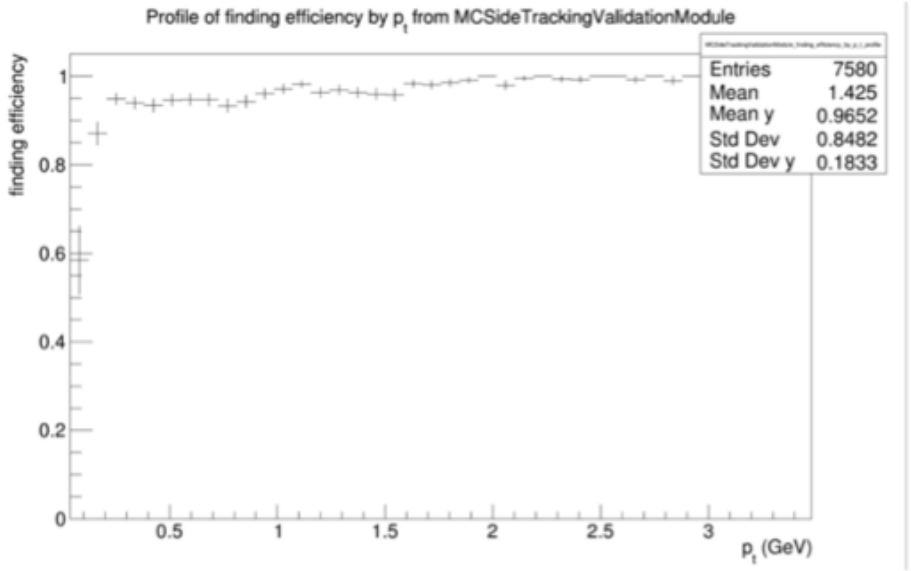
- For test purposes, I simulated one- and two-muon events with a **particle gun**, assuming a **uniform distribution** in the momentum spectrum between **0.2 and 3.5 GeV**

Track finding efficiency (μ)

M. Kuenzel



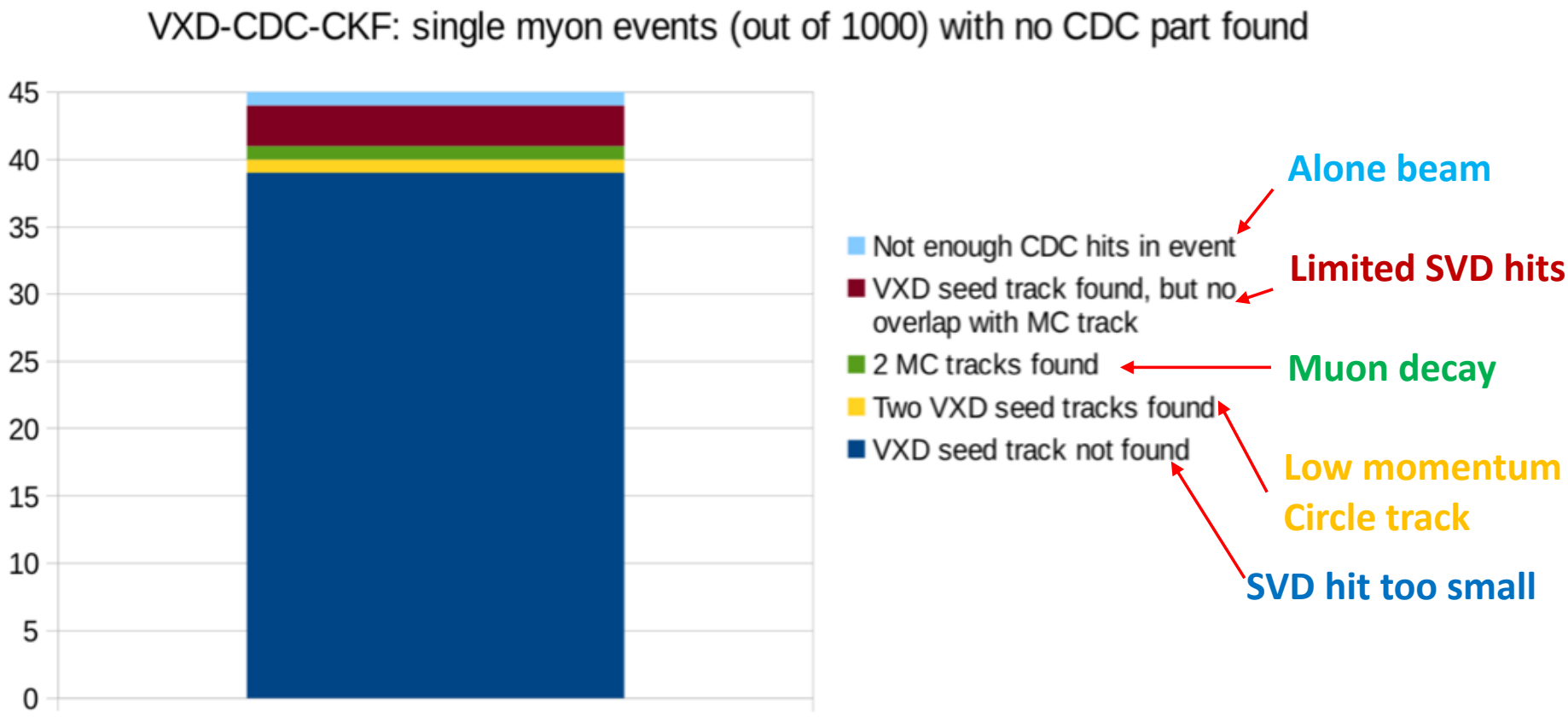
(a) Track finding efficiency vs. p_t after filter 1



(b) Track finding efficiency vs. p_t after filter 2

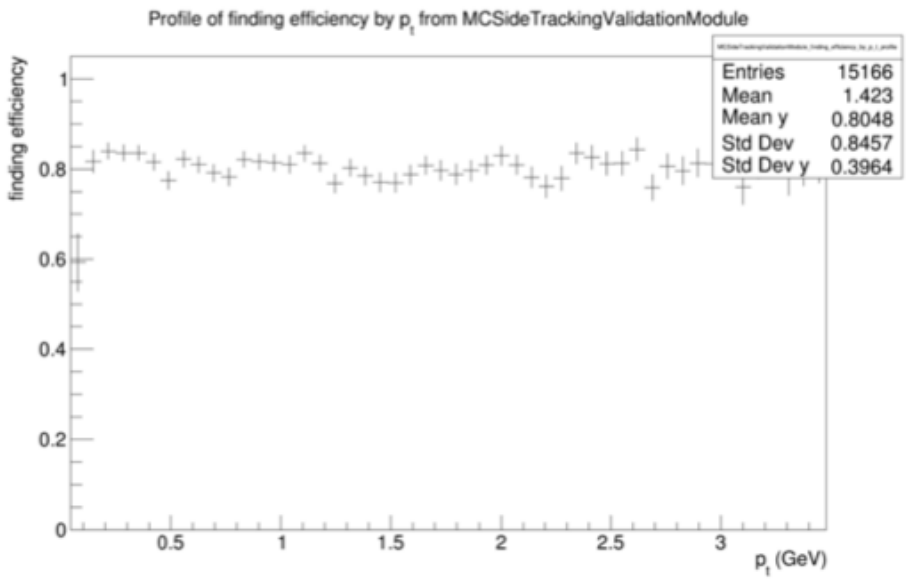
Which tracks are missing

M. Kuenzel

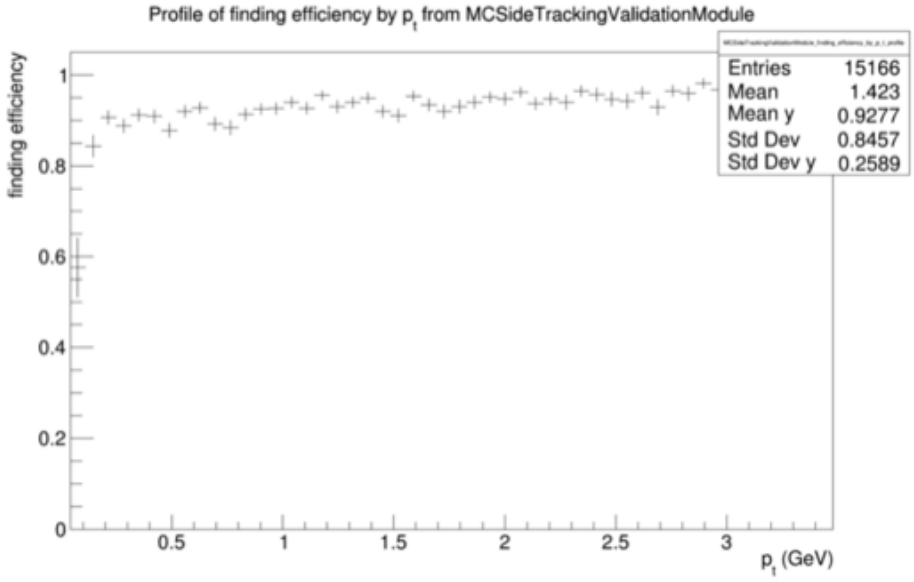


Track finding efficiency ($\mu\mu$)

M. Kuenzel



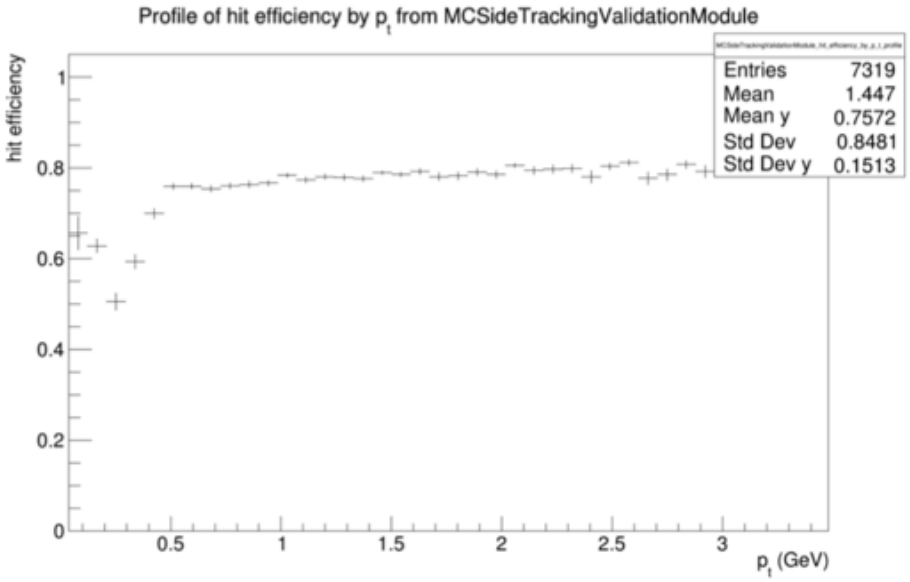
(a) Track finding efficiency vs. p_t after filter 1



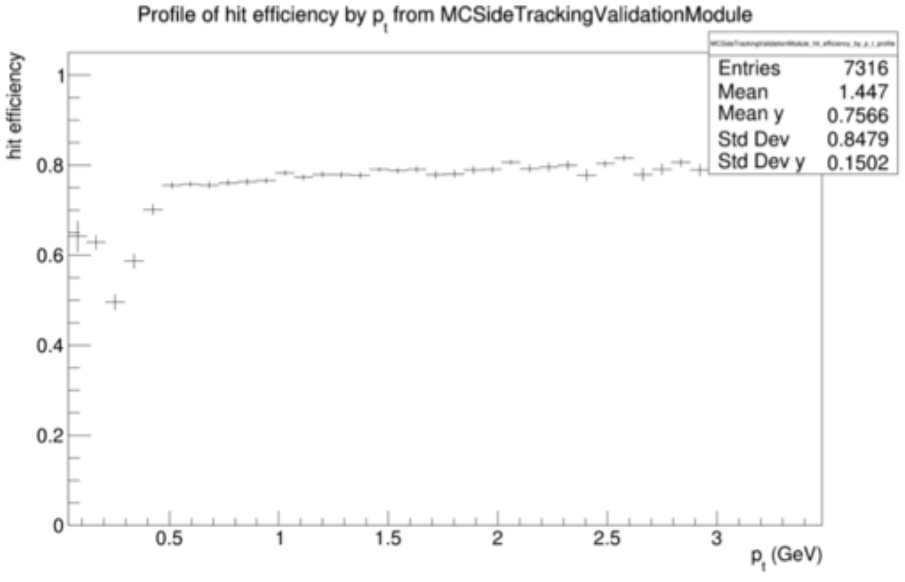
(b) Track finding efficiency vs. p_t after filter 2

Hit finding efficiency (μ)

M. Kuenzel



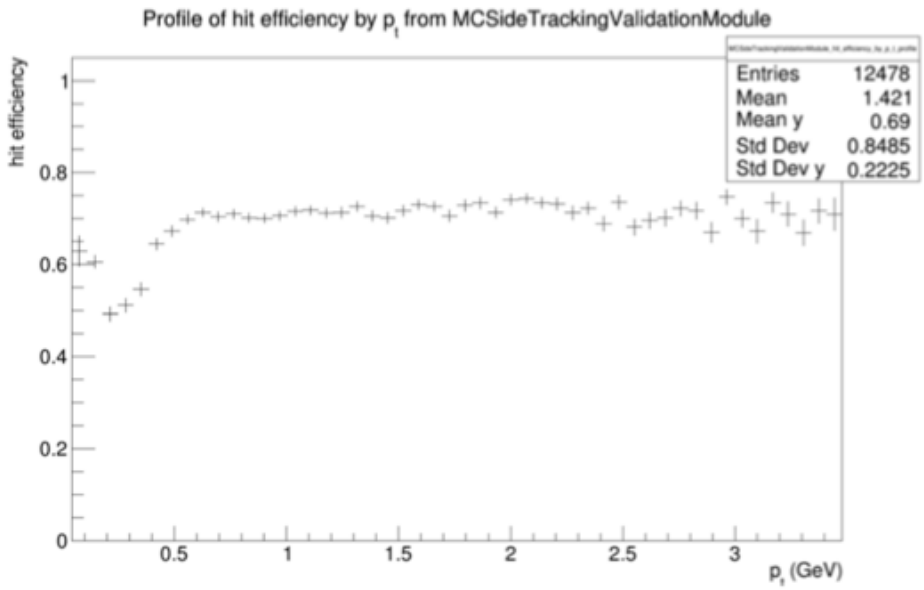
(a) Hit finding efficiency vs. p_t after filter 1



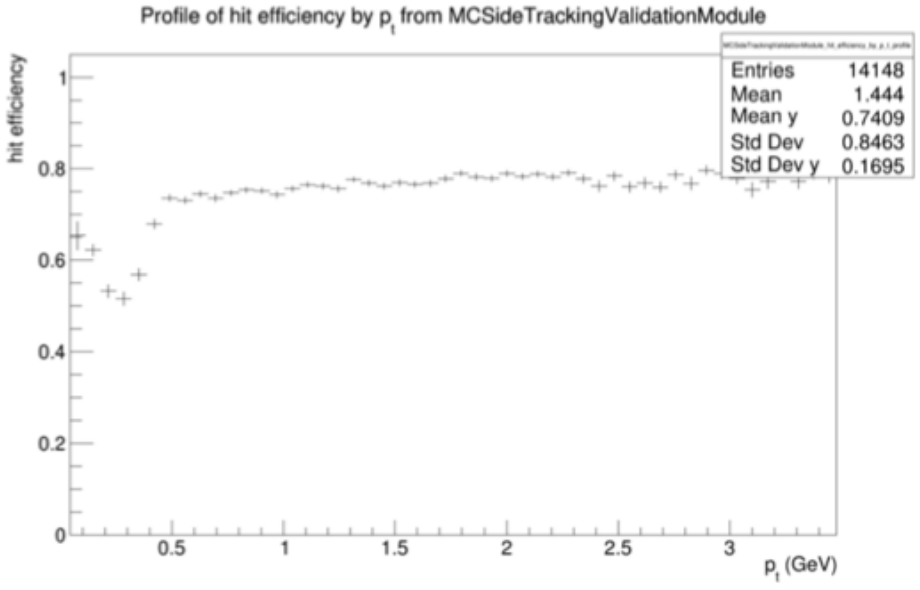
(b) Hit finding efficiency vs. p_t after filter 2

Hit finding efficiency ($\mu\mu$)

M. Kuenzel



(a) Hit finding efficiency vs. p_t after filter 1



(b) Hit finding efficiency vs. p_t after filter 2

Summary and outlook

- CDC to SVD CKF is validated roughly. Reasonable efficiency improvement is observed.
- SVD to CDC CKF implementation is on going.
- Filter 1 and 2 are implemented and first efficiency plots were produced using one- and two-muon events.

Next step:

- More parameters of CKF will be investigated.
- Filters will be optimized
- Validate the algorithmen with more background.

Thank You!

Back up

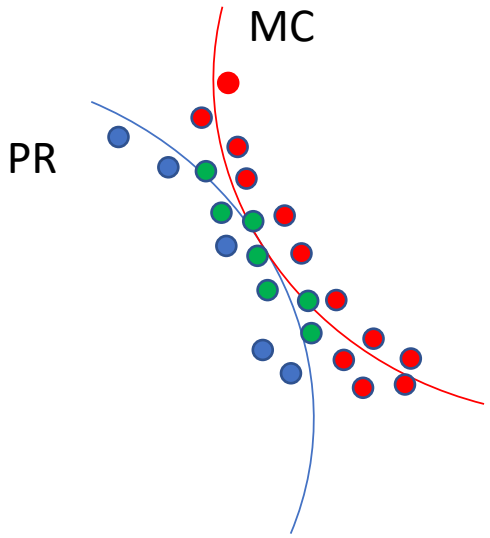
Efficiency and Purity

PR

- Matched
- Clone (1 MC 2 PR)
- Fake(bg,ghost)

MC

- Found / matched
- Merged (2 MC 1 PR)
- missing



Eff = Green/Red+Green
Purity=Green/Blue+Green