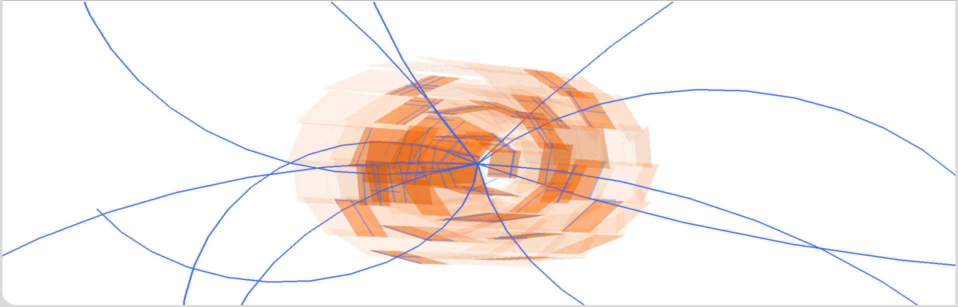
VXDTF2 6-layer tracking studies

Sebastian Racs | 18th September 2017

Institut für Experimentelle Teilchenphysik (ETP)



Overview

1. PXD SVD SpacePoint Cuts
   - Why are cuts necessary?
   - Where to cut?
   - When to cut?
   - How? New cutting module
2. VXDTF2 Parameter Variation
   - General Idea
   - Study
   - Hurdles
   - Some initial results
3. Other improvements
   - `segmentNetwork` Identifiers

PXD SVD SpacePoint Cuts

Why are cuts necessary?

- Doing naive 6 layer tracking:
  - using data with PXD reduction (reduction with `VXDTF1`)
  - running normal `add_vxd_track_finding_vxdtf2`
  - and `add_mc_matcher` with components `PXD` and `SVD`

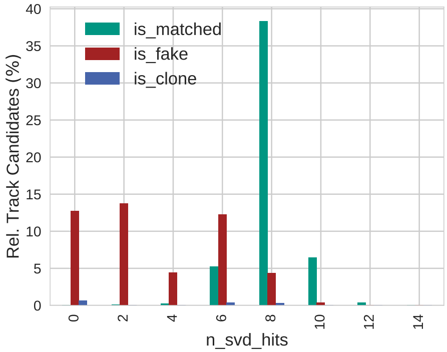


Figure: Track candidates per SVD hits with their matching status: **no cuts**

Why are cuts necessary?

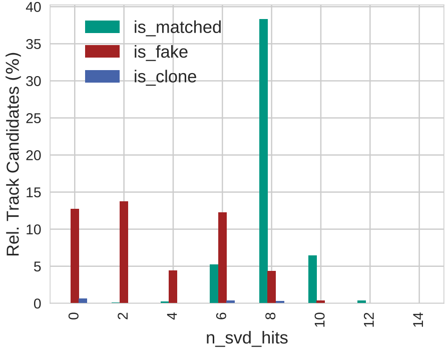


Figure: Track candidates per SVD hits with their matching status: **no cuts**

Results

- ∼13 % of tracks have 0 SVD hits; are only made up of PXD hits
- ∼30 % of tracks have less than 6 SVD hits (3 clusters), but only 0.4 % of those are actually matched

Where to cut?

Table: Figures of merit for different cuts

Cut tracks with less than # SVD Clusters	**finding** efficiency	**hit** efficiency	**clone** rate	**fake** rate
0	82.9	93.2	2.5	48.0
1	82.9	93.2	1.4	40.7
2	83.6	93.0	1.3	30.6
⇒ **3**	84.3	93.1	1.3	26.5
4	78.5	93.6	0.7	12.7

- *clone rate* and *fake rate* decrease for stricter cuts
- *finding efficiency* increases a bit to **3** but drops again for stricter cuts
- *hit efficiency* stays the same

⇒ Cut tracks with less than **3** SVD Clusters/SPs. This is also equivalent to `SVDOnly` SpacePoint requirement

When to cut?

- Do we need to cut tracks at creation time?
  - $\Rightarrow$ **No**, tracks with less than 6 SVD hits are only $\sim$7 % of the total number created
  - $\Rightarrow$ a simple module to deactivate track candidates before using the `QualityEstimator` is enough

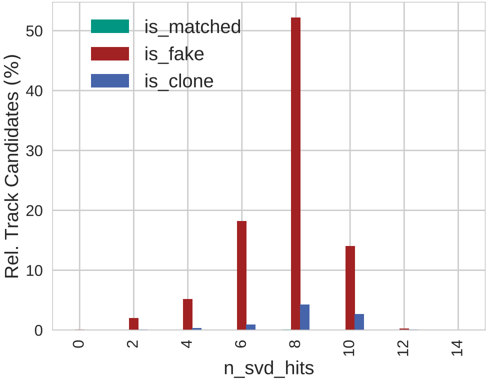
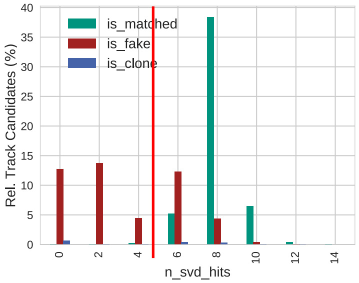
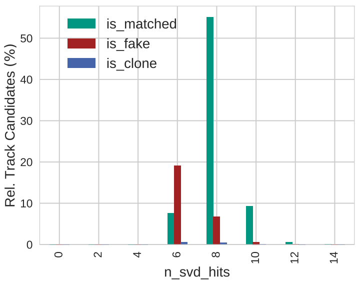


Figure: All track candidates with **no filters or overlap check** applied

How? New cutting module

- New module to run in `VXDTF2` setup before `QualityEstimator` if using PXD
- Deactivate `SpacePointTrackCandidates` with less than a minimum of `SpacePoints` of type `SVD`
- Has a parameter `minSVDSPs`, default should be set to 3





⇒ On branch `feature/pxdSVDCutModule` now

⇒ **Pull-request soon** (minor changes necessary first)

VXDTF2 Parameter Variation

General Idea

- Why does 6-layer-tracking (PXD-SVD) produce worse finding efficiencies than 4-layer (SVD-Only)?
  - Reduction with `VXDTF1` (from above): 83 %
  - Reduction with `VXDTF2` + Custom Sector Map (see Felix results): 92 %
  - vs. `SVD-Only` with ∼95 %

$\Rightarrow$ investigating `VXDTF2`'s parameters to disable filters and overlap-checks, enable additional path subsets and change the quality estimator

Study

- Require 100 % hit purity of tracks using
  - mcInfoQE/`QualityEstimatorMC`
  - + QualityIndexCutter

Comparing:

- **Default** (`SVDPXDDefaultMap`) vs. Custom (Muon) SectorMap
- Using **strict** or flexible seeding of paths
- Storing path subsets or **not**
- **Enable** and disable `SVDOverlapResolver`
- PXD-SVD 6-layer or **SVD-only**
- Data **with** and without Background and PXD Data Reduction
- **VXDCellOMat** vs. BasicPathFinder
- Replacing `SegmentNetwork` filters with `QualityEstimatorMC` filters

Hurdles

- Fixing `QualityEstimatorMC` to work with PXD and SVD
  - adding additional loops
  - had forgotten to account for 1 vs. 2 hits in SpacePoints for `estimateQuality` calculation $\Rightarrow$ had to redo all the calculations
- A lot of different possible parameters to check, not all of them work because of too much RAM or long run-times
  - using a basic validation module to write-out just finding and hit efficiency
  - instead of turning `SegmentNetwork` filters off completely, replace them with MC

Future of bugfixes and additional, optional parameters

- Still needs some cleanup before putting on stash
- Only impacts special ”debugging” features, but should still be fixed

Some initial results

- storing subset paths increases finding efficiency from 86 % to 97 %
- changing from strict to flexible seeding and turning off `SVDOverlapResolver` has only small effect
- replacing 2-hit-filter in `SegmentNetwork` also increases finding efficiency from 86 % to 97 %
- 3-hit-filter seems to have no negative effect

Outlook

- still need to further investigate my big table of calculated finding efficiencies
- some data with background and no reduction was working, but most needs to much RAM, might still be interesting
- understanding connection between PXD data reduction and finding efficiency
  - PXD data reduction with `VXDTF1` has lower finding efficiency than with `VXDTF2`, but not comparable?

Other improvements

`segmentNetwork` **Identifiers**

- `segmentNetwork` was constructed with complicated strings stored in an `unordered_map`
  - `TrackNode` and `ActiveSector` used long strings as identifiers with `getName()` function
  - `Segment` was combining two strings to create its identifier
  - `DirectedNodeNetwork` was storing Nodes in an `unordered_map`
- All of this was used by the `SegmentNetworkProducerModule`

Solution

- Replace complicated long `string` names with easy `int` identifiers
  - Using the SpacePoint datastore `getArrayIndex` function to get a unique identifier
  - For `Segment` combine two `int`s into the upper and lower halfs of another `int`

`segmentNetwork` **Identifiers**

- Additional small improvement: in `DirectedNode` reserve some space for Node vectors
  - `emplace_back` kept showing up in validations
  - reserving 10 spaces got almost rid of this

Total Result

- Module `SegmentNetworkProducer`'s run-time decreases by 50 %
  - for 1000 events, from 45.8 s to 24.1 s (while overall a bit slower)
  - 2nd slowest module in `VXDTF2` becomes 3rd slowest

Already merged into `main`!

- Pull request **679**
- solving Issue **BII-2476**