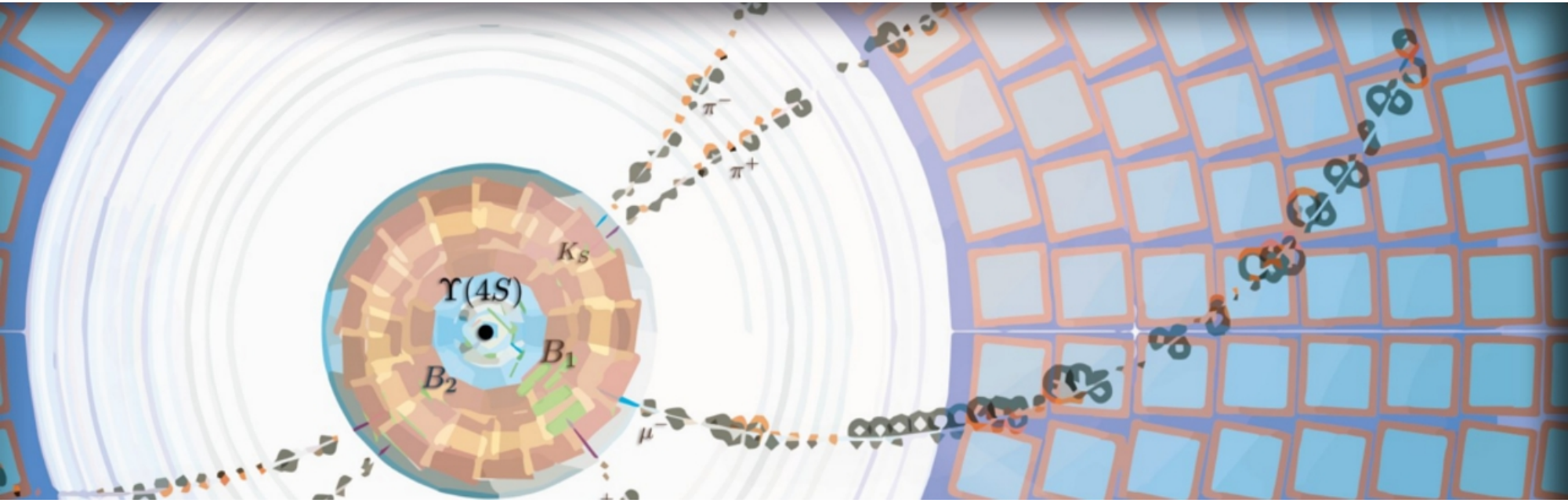
CDC track finding - Big picture and dealing with complexity.

F2F Meeting - Munich 2016



Oliver Frost
Deutsches Elektronen-Synchrotron (DESY)
2016-01-12

DESY

HELMHOLTZ ASSOCIATION

Belle II

Overview

- Review of year 2015

- Rough edges

Review of year 2015

Last years proposal – Why merge?

Double effort – local - global track finders

- Many things are implemented twice like
  - Hit objects
  - Track objects
  - Fits
  - Trajectory representations
  - Reconstruction of the z coordinate
  - A basic event display
  - Sorting of hits in a track
  - Merging of tracklets

Sharing and review

- Combining the efforts may lead to synergies.
  - Exchange of ideas, experiences and problems
  - Mutual review of code

Strong together

Mission accomplished

Combining the two complementary approaches leads to

- Higher efficiency
- Lower CPU resource consumption
- Versatility
- Replacing legacy Trasan as default finding method

Thanks to

- Viktor Trusov
- Nils Braun
- Martin Heck and Thomas Hauth for support and review

Rough edges

Products of CDC track finding

Provide finding of CDC tracks

- in normal beam events
- for the cosmics test runs
- (on HLT)

Decomposition of CDC track finders

Algorithmic components

- Track finding comprises a multitude of intertwined steps
-
- Each steps
  - has limited scope
  - should have clear boundaries
  - may carry some parameters

Elementary tasks - 1

- Translation of CDCHits
- Creations of super clusters
- Creation of clusters
- Detection of cluster backgrounds
- Construction of hit triples (facets)
- Construction of relations between hit triples
- Application of cellular automaton to extract segments
- Concatenation of segments within super layers
- Construction of axial-stereo segment pairs
- Construction of relations between axial-stereo segment pairs
- Application of cellular automaton to extract tracks from segment pairs
- Construction of axial-axial segment pairs
- Construction of axial-stereo-axial segment triples
- Construction of relations between axial-stereo-axial segment triples
- Application of cellular automaton to extract tracks from segment triples
- Concatenation of tracks

Elementary tasks - 2

- Construction of conformal hits
- Construction of axial tracks in legendre finder quad tree
  - from hits or from segments
  - on-origin or off-origin
- Merging of axial tracks
- Assoziation of stereo hits with axial tracks by
  - stereo histogramming method
  - sz hough finding
- Assoziation of left over hits to tracks
- Assoziation of left over segments to tracks
- Cleaning of tracks
- Quality assertion tools
  - Various rejection criteria
  - Pruning
  - ...
- ...

Track finding modules in the standard reconstruction chain

```
WireHitTopologyPreparer
SegmentFinderCDCFacetAutomaton
TrackFinderCDCLegendreTracking
TrackQualityAsserterCDC
StereoHitFinderCDCLegendreHistogramming
SegmentTrackCombinerDev
TrackQualityAsserterCDC
VXDTF
VXD-only
CDC-only
VXDCDCTrackMerger
```

Track finding modules - contribution for the CDC

```
*WireHitTopologyPreparer*
*SegmentFinderCDCFacetAutomaton*
*TrackFinderCDCLegendreTracking*
*TrackQualityAsserterCDC*
*StereoHitFinderCDCLegendreHistogramming*
*SegmentTrackCombinerDev*
*TrackQualityAsserterCDC*
VXDTF
VXD-only (<- Fitting)
CDC-only (<- Fitting)
VXDCDCTrackMerger
```

A more desireable reconstruction chain ?

User - fewer details please

```
*TrackFinderCDC*
```

Rational

- Consumers only want to use the one behaviour - finding tracks in the CDC
- Concise chain for better readability
- Exposing to many details and parameters may trigger users to tinker around
- Communicates one stable interface for users

A more desireable reconstruction chain ?

Expert - more details please

```
*WireHitCreator*
*SuperClusterCreator*
*ClusterCreation*
*ClusterBackgroundDetection*
*FacetCreator*
*FacetRelationCreator*
...
*TrackMerger*
*TrackQualityAsserter*
*TrackExporter*
...
```

Rational

- Higher flexibility
- Easier exchange of single steps
- Studies of impact when adding leaving out a reconstruction steps
- Prove relevance of a new element
- Stepwise training of MVA methods
- Reuse of parts with differing parameters

Contradicting tendencies

Alternatives

Each task should be a module

- Work on few object sets to achieve their goal
- Have to communicate their results over the DataStore
- Many intermediate objects in the DataStore
- Many parameters and much rearrangement (wow)

versus

Each user product should be a module

- Few selected parameters (if any at all)
- Little cluttering of the DataStore
- Aggregation of many task

Status quo

- Is a rather unsatisfactory middle ground.
- Rather complicated module event() methods
- Switch functionality on and off with parameters

Findlet concept

Solution

Allow both ? provding each task as a findlet

Requirements

Findlets must be

- easily usable as module
- easily usable as part of a module
  - allow exposition of parameters to surrounding module
  - receive event processing signals from surrounding module

Findlet interface

```
class SegmentCreatorFacetAutomaton
: public Findlet<const CDCFacet,                                  // input
                 const WeightedRelation<const CDCFacet>, // input
                 CDCRecoSegment2D>                                // output
{
  public:
  /// Add the parameters to the surrounding module
  void exposeParameters(ModuleParamList*, const std::string& prefix = "")

  /// Main function of the segment finding by the cellular automaton.
  virtual void apply(
    const std::vector<CDCFacet>& inputFacets,
    const std::vector<WeightedRelation<const CDCFacet> >& inputRelations,
    std::vector<CDCRecoSegment2D>& outputSegments
    );
};
```

Findlet usage by users - 1

The TrackFinderCDCCosmics would apply a (here slightly shortend) series of findlets:

```
void TrackFinderCDCCosmics::event()
{
  wireHitCreator.apply(outputWireHits);
  superClusterCreator.apply(inputWireHits, superClusters);
  clusterRefiner.apply(superClusters, clusters);
  facetCreator.apply(clusters, facets);
  facetRelationCreator.apply(facets, facetRelations);
  segmentCreatorFacetAutomaton.apply(facets,  facetRelations, segments);
  segmentFitter.apply(segments);
  segmentOrienter.apply(segments, orientedSegments);
  segmentPairCreator.apply(orientedSegments, segmentPairs);
  segmentPairRelationCreator.apply(segmentPairs, segmentPairRelations);
  trackCreatorSegmentPairAutomaton.apply(segmentPairs,
                                         segmentPairRelations,
                                         tracks);
  trackOrienter.apply(tracks, orientedTracks);
  trackExporter.apply(orientedTracks);
}
```

(all variables interpreted as members.)

Findlet usage by users - 2

Get the whole prepared functionality with

```
path.add_module("TrackFinderCDCCosmics")
```

Findlet usage inside a module

In a steering file experts may manually compose:

```
path.add_module("WireHitCreator")
path.add_module("SuperClusterCreator")
path.add_module("ClusterRefiner")
path.add_module("FacetCreator")
path.add_module("FacetRelationCreator")
path.add_module("SegmentCreatorFacetAutomaton")
path.add_module("SegmentFitter")
path.add_module("SegmentOrienter",
                segments="orientedSegments",
                segmentOrientation="symmetric")
path.add_module("SegmentPairCreator",
                segments="orientedSegments")
path.add_module("SegmentPairRelationCreator")
path.add_module("TrackCreatorSegmentPairAutomaton")
path.add_module("TrackOrienter",
                tracks="orientedTracks",
                trackOrientation="symmetric")
path.add_module("TrackExporter", tracks="orientedTracks")
```

and tinker whatever.

Conclusion

Implementation

- Findlets publish the objects they operate on as template parameters
- Communication of this desired objects with the DataStore with a simple FindletModule adapter
- Status : Picked apart and recomposed the TrackFinderCDCCosmics with Findlets
- Concept proven to be sound

Decision

- Proceed with this decomposition
- or keep the status quo?

Additional pain point

- Still may different implementations of the validation
- Yet to define a standard way to retrain mva methods
- Out of sync python code and examples.
- Integration of the RecoTrack model