A Kalman filter track-fitter in the MarlinTPC framework

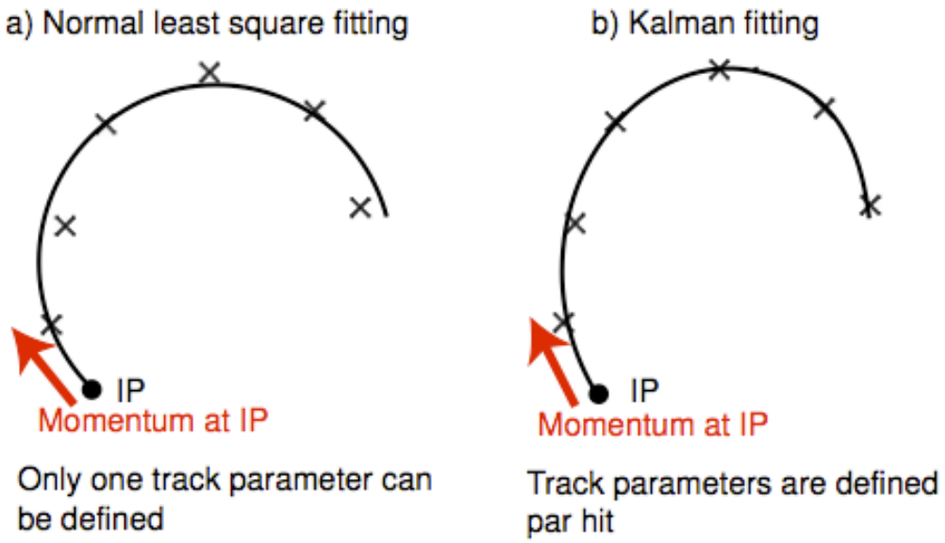
Katsumasa Ikematsu (KEK/IPNS)

LC-TPC MarlinTPC work meeting

24/11/2009 @DESY-HH

Disclaimer

- Skip an explanation for the principle of Kalman filter
  - but there is a good document and a good slide (talk by Keisuke)
    - http://www-jlc.kek.jp/subg/offl/kaltest/doc/ReferenceManual.pdf
    - http://www-jlc.kek.jp/subg/offl/kaltest/doc/kalman.pdf



- Our implementation depends on 2 external C++ class libraries (should be loaded via $MARLIN_DLL)
  - Kalman filter engine: **KalTest** (**Kal**man filter **T**racking **E**nvironment w/ **S**tandard **T**est suits)
    - http://jlccvs.kek.jp/cgi-bin/cvsweb.cgi/KalTest/?cvsroot=CDC
  - User defined application-specific detector class library: **KalDet**
    - http://jlccvs.kek.jp/cgi-bin/cvsweb.cgi/KalDet/?cvsroot=CDC

Disclaimer (cont'd)

- We brought a Kalman filter track fitter of Marlin implementation but still need to organize the source codes before committing into MarlinTPC svn trunk
  - http://www-jlc.kek.jp/~ikematsu/tpc/marlintpcweek-nov09/TrackFitterKalmanProcessor.h
  - http://www-jlc.kek.jp/~ikematsu/tpc/marlintpcweek-nov09/TrackFitterKalman.h
  - http://www-jlc.kek.jp/~ikematsu/tpc/marlintpcweek-nov09/TrackFitterKalmanProcessor.cc
  - http://www-jlc.kek.jp/~ikematsu/tpc/marlintpcweek-nov09/TrackFitterKalman.cc
  - http://www-jlc.kek.jp/~ikematsu/tpc/marlintpcweek-nov09/TrackFitterFactory.cc

  - Treatment of (at the moment force to eliminate) double hits which are associated with a pad row (in `TPCSeedTracks`)
  - Instantiation of TFile and TNtupleD class objects in the processor -> Should use AIDA

- Single module performance (results for geometric mean of resolution in each pad rows) was checked and it's consistent with our ROOT-based analysis program

Why Kalman filter tracking ?

- Need to take into account the inhomogeneous field (not only the LP1 condition but also the ILC condition due to Anti-DID field)
  - Track model can change site to site which allows B-field variation along a particle trajectory!

- No need for calculation of Matrix inversion
  - In general the matrix inversion takes time under the space point measurements of ~ 200 sampling!

- Track connection between sub-detectors (multiple scattering and energy loss in the boundaries) can be considered

- Beneficial to calculate geometric mean resolution by using “Inverse Kalman filter” (in/out target row hit)

KalTest

- Kalman filter Tracking Environment w/ Standard Test suits
  - A Kalman Filter Package written in C++. It is based on ROOT and provides basic libraries for track fitting with Kalman filter technique. The package is distributed with some sample test programs to illustrate their usage.

- Kalman filter library features
  - **KalLib**: Kalman filter defines a generic procedure and has a much wider scope than track fitting. This implies necessity for a library of generic abstract base classes that implement the generic algorithm of the Kalman Filter.
  - **KalTrackLib**: By inheriting from the generic base classes in KalLib and implementing their pure virtual methods for track fitting purpose, we then realize a Kalman-filter-based track fitter library. However, KalTrackLib should not depend on any particular track model or shape or coordinate system of a measurement layer according to the above guideline.
  - **GeomLib**: hence separate out geometry classes that provide track model (helix, straight line, ...) and surfaces (cylinder, hyperboloid, flat plane, etc.) as a geometry library.

  - Since the tracking system of a collider detector usually consists of various components such as a vertex detector (VD), an intermediate tracker (IT), a central tracker (CT), etc., which have different shapes and coordinate systems, the software package for Kalman-filter-based track fitting should be able to accommodate a measurement layer with any shape and/or coordinate system. Considering possible extrapolation of a track to an outer tracking device such as a muon detector, it is also desirable that the package can handle site-to-site change of the magnetic field.

KalTest (requirement for user)

- Require minimum number of user-implemented classes to the following three

  - **MeasLayer**: a measurement layer that multiply inherits from an abstract measurement layer class `TVMeasLayer` and a shape class derived from `TVSurface` in GeomLib.
  - **KalDetector**: an array class derived from `TVKalDetector` that holds the user-defined MeasLayers with any shape and/or coordinate system. Notice that this also defines material distributions in the tracker.
  - **Hit**: a coordinate vector class as defined by the MeasLayer, which inherits from `TVTrackHit`.

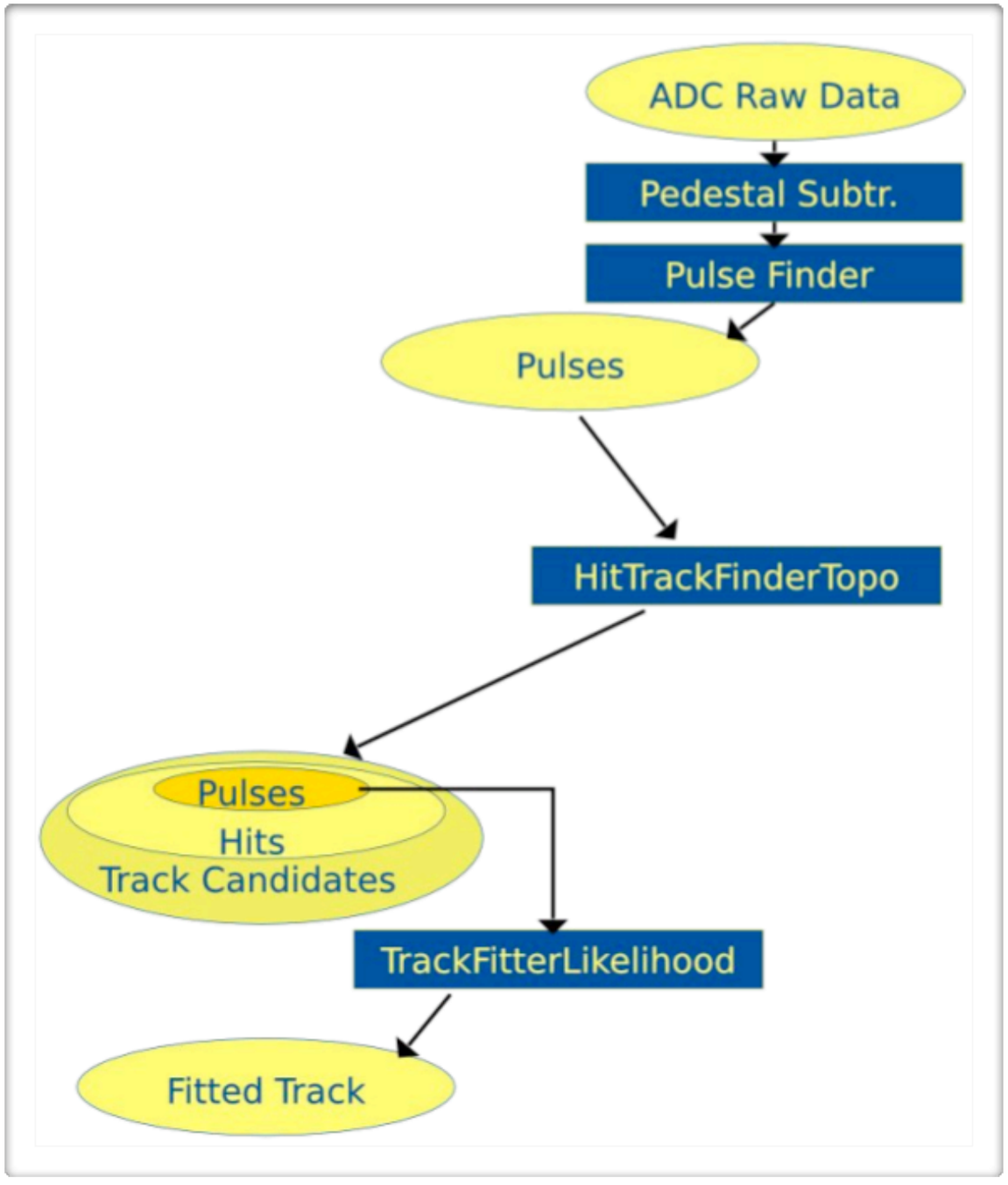
KalDet

- provides user-defined classes required by KalTest
- **kern**: application-specific abstract classes
- **lctpc/lp1gem**: provides LC-TPC LP1-JGEM modules' "KalHit", "MeasLayer", and "KalDetector"
- Can put your detector detector for example, **othertpc/medi-tpc** etc.

MarlinTPC reconstruction chain

- MarlinTPC: Set of processors for LC-TPC reconstructions and analyses (incl. simulation)
- List of processors (needed for Asian GEM module data reconstruction)

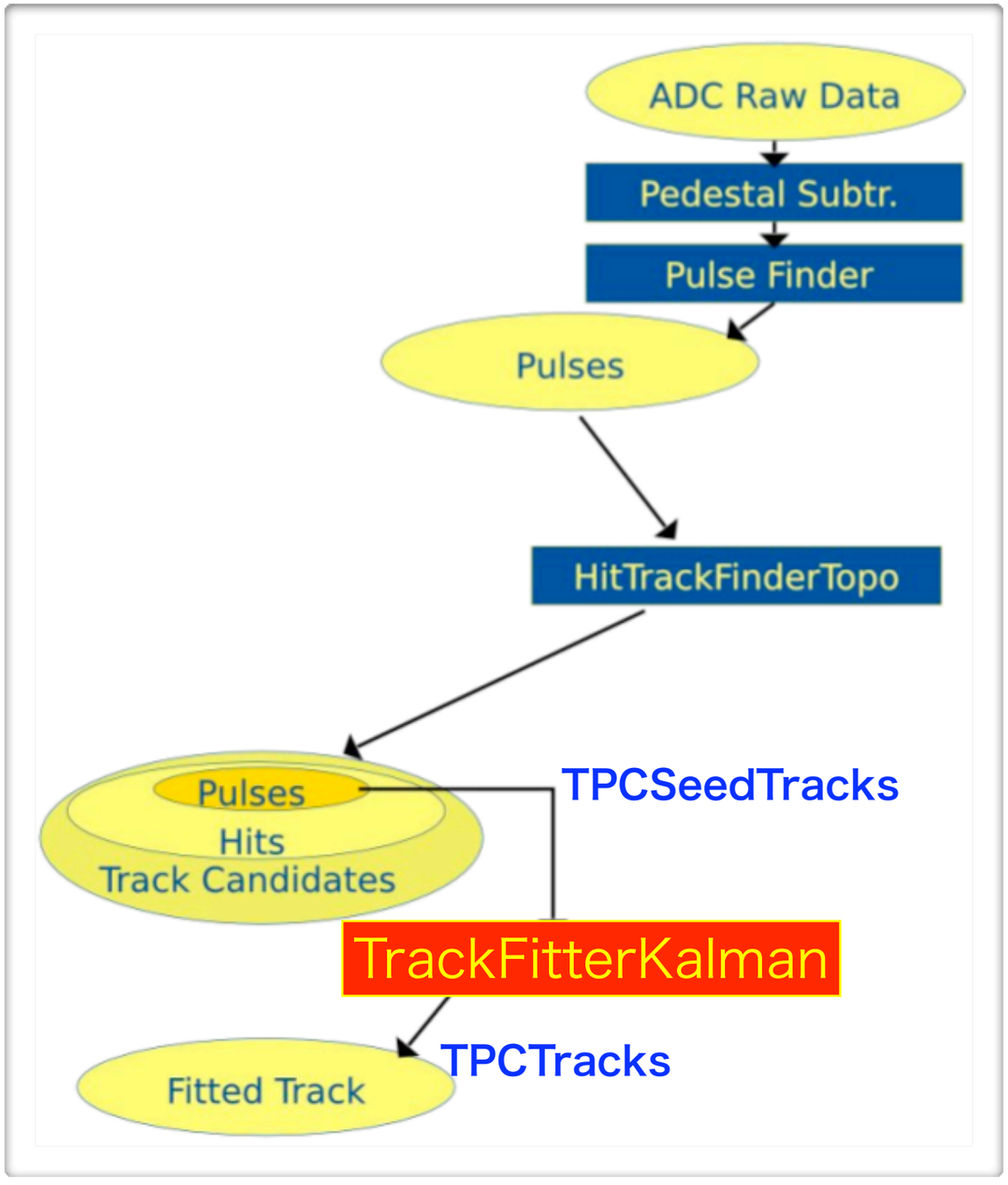
  - ConditionsProcessor (TPCConditions, TPCPedestal & TPCChannelMapping)
    - Provides access to conditions data transparently from LCIO files or a databases, using LCCD
    - At the moment TPCChannelMapping only
  - TrackerRawDataToDataConverterProcessor
    - Converts the TrackerRawData to TrackerData without processing the values
  - ADCPulseConverterProcessor
    - Convert zero-suppressed ADC raw data to pulses
  - ChannelMapperProcessor
    - Changes cellID to map from hardware channel numbers to sowftware/logical channel numbers



MarlinTPC reconstruction chain (cont'd)

- List of processors (needed for Asian GEM module data reconstruction)

  - HitTrackFinderTopoProcessor
    - Calculates TrackerHits from TrackerPulses
  - TrackSeederProcessor
    - Calculates seed track parameters from in the TrackerHits in the track candidates collection
  - **TrackFitterKalmanProcessor**
    - **Our work!!**
  - HepRepOutputProcessor
    - Generates a HepRep Output file for a HepRep based event display program
    - HepRep XML file which can be displayed e. g. with Wired/JAS3 or HepRApp (.jar)
  - LCIOOutputProcessor
    - Writes the current event to the specified LCIO output file
    - Needs to be the last Active Processor
    - Split output file if size in 1992294 kB exceeds



Excerpt from our steering XML file

```
<marlin>
<!--##########################################
    #                                        #
    #     steering file for reconstruction   #
    #                                        #
    ##########################################-->

 <execute>
  <processor name="MyAIDAProcessor"/>
  <processor name="MyConditionsProcessor"/>
  <processor name="MyTrackerRawDataToDataConverterProcessor"/>
  <processor name="MyADCPulseConverterProcessor"/>
  <processor name="MyChannelMapperProcessor"/>
  <processor name="MyHitTrackFinderTopoProcessor"/>
  <processor name="MyTrackSeederProcessor"/>
  <processor name="MyTrackFitterKalmanProcessor"/>
  <processor name="MyHepRepOutputProcessor"/>
  <processor name="MyLCIOOutputProcessor"/>
 </execute>

 <global>
  <parameter name="LCIOInputFiles"> readout-7049.000.slcio </parameter>
  <!-- limit the number of processed records (run+evt): -->
  <parameter name="MaxRecordNumber" value="3" />
  <parameter name="SkipNEvents" value="0" />
  <parameter name="SupressCheck" value="false" />
  <parameter name="GearXMLFile"> gear_LP_TPC_GEM_7module.xml </parameter>
  <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> DEBUG  </parameter>
 </global>

...
...

 <processor name="MyTrackFitterKalmanProcessor" type="TrackFitterKalmanProcessor">
  <!-- The the name of the input collection of track candidates (default: TPCSeedTracks) -->
  <parameter name="InputSeedTracks" type="string" lcioInType="Track">TPCSeedTracks </parameter>
  <!-- The name of the output collection with the fitted tracks (default: TPCTracks) -->
  <parameter name="OutputTracks" type="string" lcioOutType="Track">TPCTracks </parameter>
  <!-- if not 0 the output hits collection is set transient (default: 0) -->
  <parameter name="SetOutputTransient" type="int">0 </parameter>
 </processor>

...
...
```