Offline Tracking Status & Plans

Norman Graf (SLAC)

HPS Collaboration Meeting @ JLab
May 30, 2019

The Global View

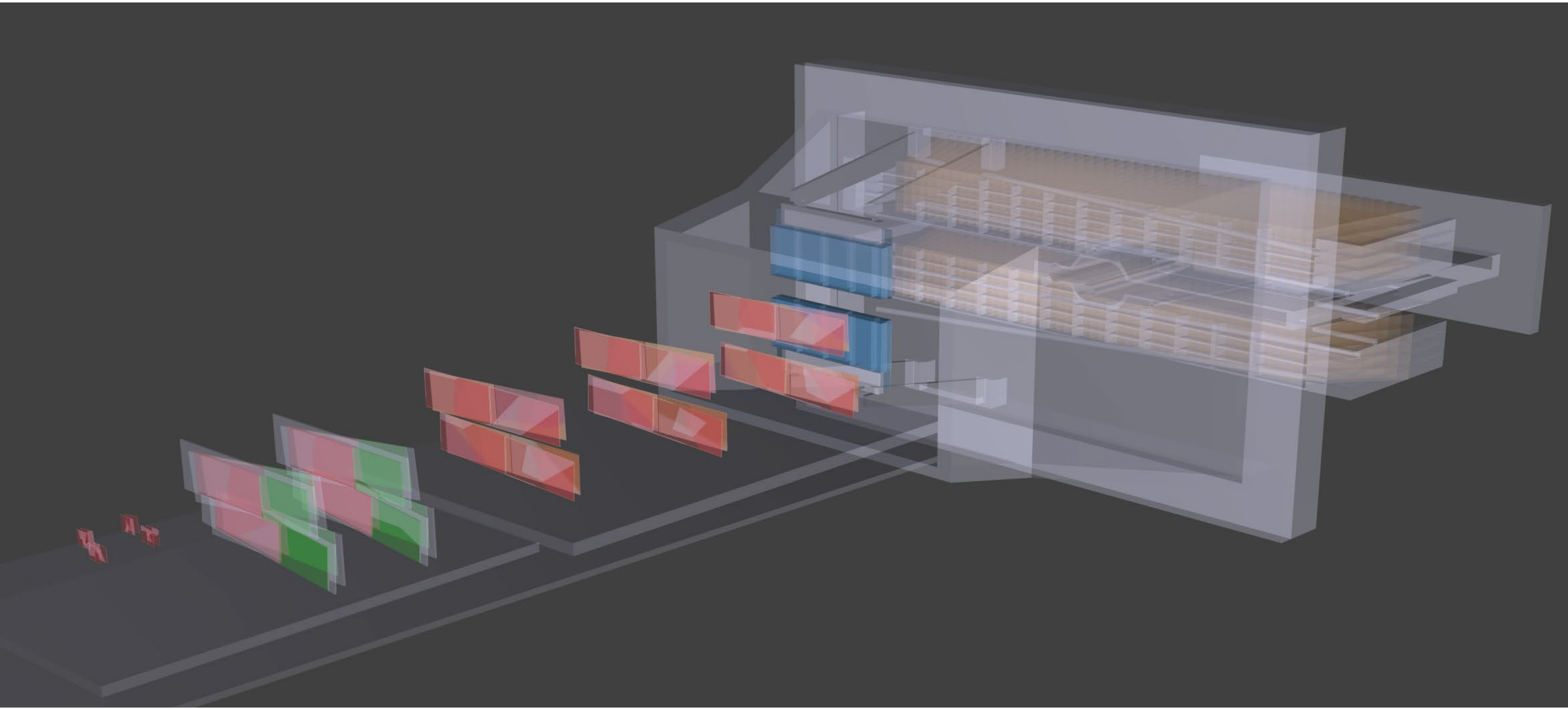
- Glass half full:
  - The tracking and vertexing works…
    - Multiple theses
    - Physics publication
- Glass half empty:
  - We could be doing (much) better
    - Better & faster simulation
    - Better & faster calibration / alignment
    - Better & faster reconstruction
    - Better tracking efficiency
    - Improved track and vertexing resolutions
- What do we NEED for this run NOW?

Tracking Triage

- New Detector
  - Correctly handle new L0 sensor
  - Survey positions for all subdetector elements, including magnet, target(s), etc.
- New Tracking Strategies
- Run Plan for Alignment/Calibration
  - Field-off, FEE, SVT wire target, …
- Manpower to analyze the data, align/calibrate the detector, improve the software.

New SVT Layout

- Added new Layer0
- Swapped Layer0 sensor into “slim” Layer1



Handling Layer0 Sensor

- The new Layer0 sensor has split strips, read out from both sides.
- Had previously simulated these sensors by creating two sensors, similar to layers 4-6.
- Had expected that we could handle real data with a simple modification to the DAQ map which assigns electronics readout channel number to silicon strip.
- Realized just recently that this will not work for both MC and data.
- Resolution of this involves re-architecting some of the base classes which handle the sensors and electrodes.
- Manpower split between SVT hardware and software.
  - Hardware taking priority.
- Recognized as a critical path item.

Detector Survey

- Need to incorporate survey information into a new 2019 Detector description.
  - All SVT sensors
  - Magnet location and orientation wrt (0,0,0)
  - Target(s) and field-off target locations

Software CPU Performance

- Our tracking software is SLOW!
  - Not a critical issue for 2015/6, definitely an issue now.
  - Have detailed profiling data, but there has been no appreciable action to-date
  - Overall CPU budget dominated by tracking, primarily track-finding/fitting, followed by raw hit-fitting
- Fitting SVT readout samples to determine hit time and pulse height
  - Currently using generic minuit fit
  - Need to evaluate possible gains from a dedicated fitter
  - Fit once in pass0, don't refit in later passes.
  - Effort started with rotation student at SLAC, will be continued.
  - If C++/root-based approach is faster/better, may implement an intermediate step in processing
  - Split large evio file→fit SVT t0/amplitude→ write smaller lcio files→ reconstruct with hps-java

Track Finding

- By default we will continue to run the SeedTracker pattern recognition, which creates 3D hits from pairs of axial & stereo strip clusters.
- Will want to run the StrategyBuilder to create a new set of track-finding strategies that include Layer0 and “slim” Layer1.
- Fall-back is to utilize 5-hit tracking based on layers 2-6.

- Working on alternative track-finding algorithms for full production and final analysis.

Plan for alignment-related activities: data requirements

- **Data for alignment**
  - Alignment with straight tracks always more advisable due to the (coarse) available precision of the magnetic field mapping, especially in the fringe field regions
  - In 2016 one run only without magnetic field was taken, and the end of the data taking
    - Too few data
    - Not representative enough for the whole data taking period
  - At least 3-5 times more statistics would be desirable, if possible spread along the full data taking (e.g.: one stock of data at the beginning, one along the run, one at the end of the data taking)

- **Data for calibration**
  - FEE tracks for momentum calibration (but not really sure dedicated trigger runs are necessary)

Software status: reconstruction and interface to alignment

- Reconstruction: two critical issues
  - **New entry: layer 0**
    - Integrate in the geometry (done but beware: the chosen geometry must be a steady version common to REC and MC, otherwise it won't be possible to train the alignment machinery on MC data)
    - Adapt the Millepede framework to match the new layout with layer0
      - Extract the new information provided by the tracking (hits on the new layer)
      - Provide Millepede with an additional layer for offsets calculations
        - » Provide/check new coordinates, derivatives, tranformations between local/lab reference systems
      - Change accordingly the **BuildMillepedeCompact** class which translates the offsets found my MP into a new compact.xml file
    - Revise the **DetectorConverter** class for geometry preparation and visualization (e.g. DrawLCDD.py on lcdd files)
  - **Revision of straight track reconstruction code**
    - Changes are due for the insertion of the new layer
    - Remember that we always got different outputs for the best aligned geometry if using straight or curved tracks!
      - Still needs to be fixed and carefully tested

A. Filippi

Software status: alignment tools

- hps-java has been modified in order to provide directly Millepede with a binary input upon reconstruction
  - Before: an ascii file was written and read by a python procedure preparing the input for MP (very time and resource consuming, BUT all the refits and intermediate steps following GBL application could be under control at each stage)
  - Now: the binary file is written directly by hps-java
    - Same source as reconstruction output
    - Tested on 2016 *curved tracks*, it works
    - *Never tested on straight tracks*
    - To be tested **carefully** with the additional layer (check consistency, correspondences, …)
    - *Note*: there is no backward compatibility between the two procedures (so we must get it fully working as it is now)
    - Output format: root file
      - Adapt existing macros
      - Check if all needed information is available, add missing items

Summary: to-do list for alignment readiness

- **Reconstruction**
  - Revise straight tracks reconstruction: procedure and output format
  - Check output for Millepede processing with the additional layer0
  - Make sure of consistency of all information to be provided to Millepede in the binary file
  - Revise DetectorConverter package
  - Revise functionality of geometry visualization tools (based on SLIC: so the geometry *must* be consistent in rec and simulation)
- **Alignment software**
  - Check Millepede interface for data readout (one more layer) and input to the minimization program
  - Tune rootfile output
    - Additional histograms for new layers
    - Check is some important information is missing
    - Revise macros for the visualization of residuals, momentum spectra, radiographies, etc.
  - Modify the BuildMillepedeCompact class to write the compact.xml file corresponding to a new geometry

A. Filippi

Helpers welcome! (as usual)

Alignment Moving Forward

- Include beam spot (and ECal?) into alignment procedure using single-tracks
- Include vertex constraint for multiple track events

- Couples top and bottom halves of detector
- Constrains weak (momentum) mode

Track Reconstruction Software

- Track finding and fitting were adapted from software developed for generic collider detectors
- Adoption of this software allowed rapid development during the design phase of HPS but required a few compromises
  - Use of a generic geometry definition and pattern-recognition system.
    - Fast for development, not optimized for production.
  - Rotation of our coordinate system to spoof a solenoidal field
  - Use of track parameters not natural for a fixed-target geometry.

Pattern Recognition

- Possible improvements:
  - Improved axial/stereo matching (L4-L6)
  - Improved and/or more strategies using 3D points
  - Cluster-seeded tracking
    - ECal cluster position and energy define a trajectory which originates from the beam-spot (HPS Note 2015-006).
    - Find tracks consistent with that hypothesis.
  - Implement pattern recognition based on 1D strip hits.
    - No "ghost" hits, or parallax issues
    - Could see increased efficiency by not requiring hits in both axial and stereo layers per station.
    - Fits naturally into a Kalman Filter approach.

Kalman Filter Status

R.P. Johnson

May 25, 2019

Kalman Filter Objective

- Develop a new pattern recognition program that
  - Never makes use of “3-D hits”, for improved efficiency.
  - Makes use of the full non-uniform field map.
  - Uses statistically meaningful error estimates for picking up hits.
- The objective has *not* been to replace the existing GBL fit
  - In principle the GBL and Kalman fits should be more-or-less equivalent.
  - However, in the process of doing this we did discover that the GBL fit assumes a uniform field, which may have some disadvantage.
  - We also uncovered a serious bug in the HPS field map files (which was corrected some time ago).
  - The Runge-Kutta integration code written for the Kalman Filter implementation and testing was adapted by Miriam to extrapolate tracks to the target and to the electromagnetic calorimeter.

Existing Code

- SeedTrack: does a simultaneous linear fit to a circle and line (helix approximation), to generate "guess" helix parameters and covariance for starting the Kalman filter.
  - Requires at least 3 stereo hits and 2 axial hits.
- KalmanTrackFit2: executes the Kalman filter and smoothing steps for a given set of hits.
  - Starts in the 4th or 5th layer and filters toward the target (in anticipation of the likely direction of a pattern recognition algorithm).
  - Then it restarts at the target end, filters to layer 6, and smooths back to the target.
- KalmanPatRecHPS: first attempt at a combinatorial pattern recognition based on the SeedTrack and Kalman code.
- KalmanDriverHPS and KalmanInterface: code by Miriam to interface with the HPS Java programs.

Kalman Filter Fitting Code Status

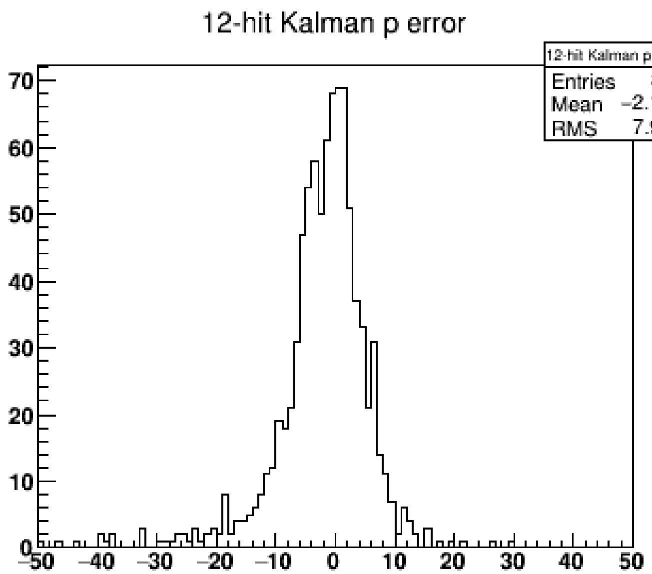
- The mathematics of the Kalman Filter code has been thoroughly tested by means of an idealized simulation:
  - Complete geometry of ½ of the HPS silicon tracker, using the surveyed positions and angles.
  - Runge-Kutta integration of a simulated particle through the HPS field.
  - Gaussian smearing of Si intersection points by 6-micron resolution.
  - Gaussian multiple scattering in each silicon plane.
- The pull distributions in each layer are very close to being normal (except for some skew in Layer 6).
- The distributions of helix-parameter errors (relative to MC truth), divided by error estimates, are normal.
- The chi-squared of the helix parameters calculated from the full 5×5 covariance matrix is distributed correctly for a chi-squared distribution with 5 d.o.f.

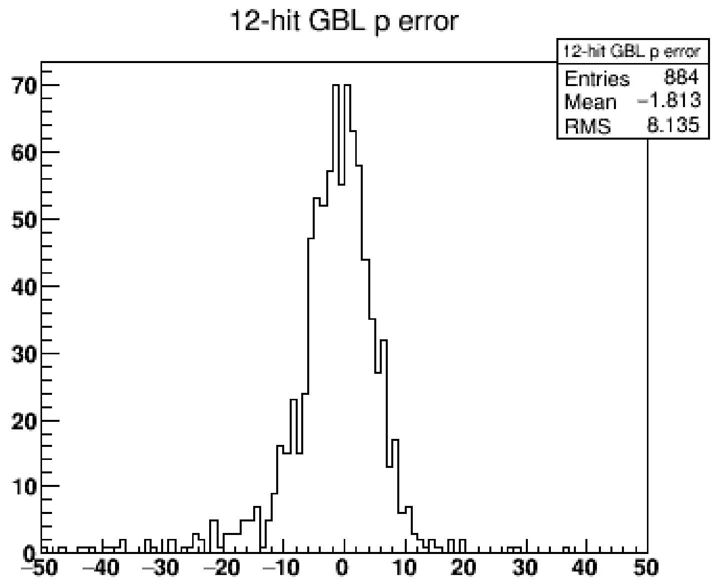
Mathematical Issues

- The fit chi-squared, summed over the 12 layers, has a mean of 12, as in a chi-squared distribution for 12 d.o.f., but the rms of the distribution at ~10 is significantly larger than the $\sqrt{24}$ expected for a chi-squared distribution of 12 d.o.f.
  - To understand this better, a toy Kalman filter for a 2-D line fit with multiple scattering was written, and it showed the same behavior.
- The pull distributions in Layer 6 (11^{th} and 12^{th} planes) are noticeably asymmetric. The asymmetry goes away if the magnetic field is made uniform.
  - Putting dummy planes in between layers 5 and 6, to take smaller steps in the field, does not help.
  - We're still looking to make sure there isn't an error in the coordinate transformations used to handle field non-uniformities.

Kalman Filter in HPS Java

- The interface code originally written by Miriam runs the Kalman fitting code on exactly the same hits as used by GBL.
  - The initial guess can be the GBL fit or can be generated from SeedTrack.
  - Histograms are filled to compare the results between Kalman and GBL.
- With 12 hits the Kalman mean χ^2 at 26 is about double that of the GBL and has a larger tail.
- The two programs agree quite well on the rms kink in each layer.
- Comparing fit results against MC truth is similar between the two:





Plans

- There is still some concern about the asymmetric pull distributions in Layer 6, so we will investigate more the code used to handle the non-uniform B field.
- Work will continue on comparing with Monte Carlo truth, hoping to compare at the individual hit level.
  - This will be especially important for pattern recognition development.
- Integrate the pattern recognition code into HPS Java and start testing and tuning it on realistic MC events.

Manpower

- Many of the principal developers of the tracking/vertexing software have moved on
- Opportunities abound for individuals or institutions to contribute, either improving existing software or developing/implementing new code.

Summary

- Current code and algorithms are working, but…
- Correct handling of new Layer0 sensors requires some code re-architecting: critical path item.
- Need to define new Detector ASAP.
- Need manpower for alignment/calibration.
- Major changes are unlikely before the start of the run, but current track finding/fitting should be good enough for data quality monitoring.
- Lots of ideas for improvement.
- Great opportunities for new contributors.

Longer Term Tracking Goals

- Improve the readout pulse-shape fits
  - Enable next passes to start from existing fits
- Improve pattern recognition
  - Refine strategies, implement cluster-seeded algs.
  - Implement strip-based algorithms (e.g. Kalman)
- Improve fitting
  - Correctly handle scattering and energy loss
  - Include full fits at multiple track states
    - Allow for true residuals to be calculated/monitored
- Reduce output size (drop unnecessary collections)
- Speed everything up.