SoLID Tracking

Weizhi Xiong

07/28/2016

Outline

- SIDIS tracking with one time sample
- Kalman Filter as tracking finding algorithm
  - Seeding
  - Track following
  - Final selection for output
- Preliminary results
- Conclusion and what can be improved

SIDIS Tracking with One Time Sample From APV

- Data size limitation allows only one time sample from the APV, if we aim for 100k Hz
- With only one time sample, we cannot apply noise cut, the most effective cut so far for rejecting out-of-time noise (how this cut behave under trigger jitter and noise still need to be tested)
- Threshold cut on APV signal amplitude becomes essentially the only tool to suppress noise
- Fairly high occupancy and hit multiplicity after threshold cut (ADC = 120):

	GEM 1	GEM 2	GEM 3	GEM 4	GEM 5	GEM 6
Occupancy	2.4%	9.4%	4.0%	2.4%	1.9%	1.4%
Hit Multi.	387	4591	1751	1025	427	408

Hit multiplicity contains false hits
number will go up if consider 20 GEM sectors (currently assume 30)
Lead to large amount of combinations ~10^{17} to 10^{18} currently

SIDIS Tracking with One Time Sample From APV

- At this level, Progressive tracking doesn't work
- Large angle can still hold up thanks to the LAEC
- Forward angle breaks down first due to high hit multiplicity and lack of enough support from downstream detectors (ECal, MRPC… are about 3m away downstream from the last GEM)
- Execution time also increase dramatically

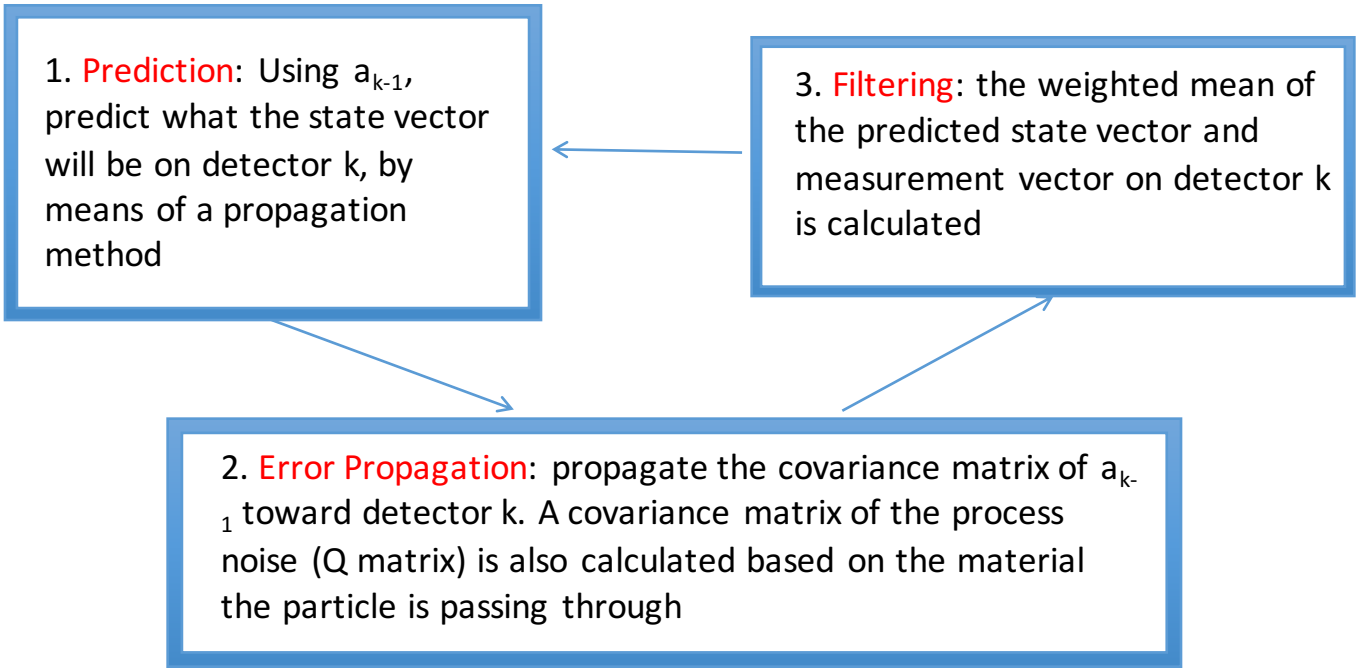
Previous result with Progressive tracking and one time sample from APV (Forward Angle)
Single electron signal track

	Zero track	Single track	Multi track
Efficiency	2.1%	38.4%	59.5%

	0	1	2	3	4	5
# of misidentified hit per track	28.7%	8.6%	2.8%	2.6%	55.4%	1.9%

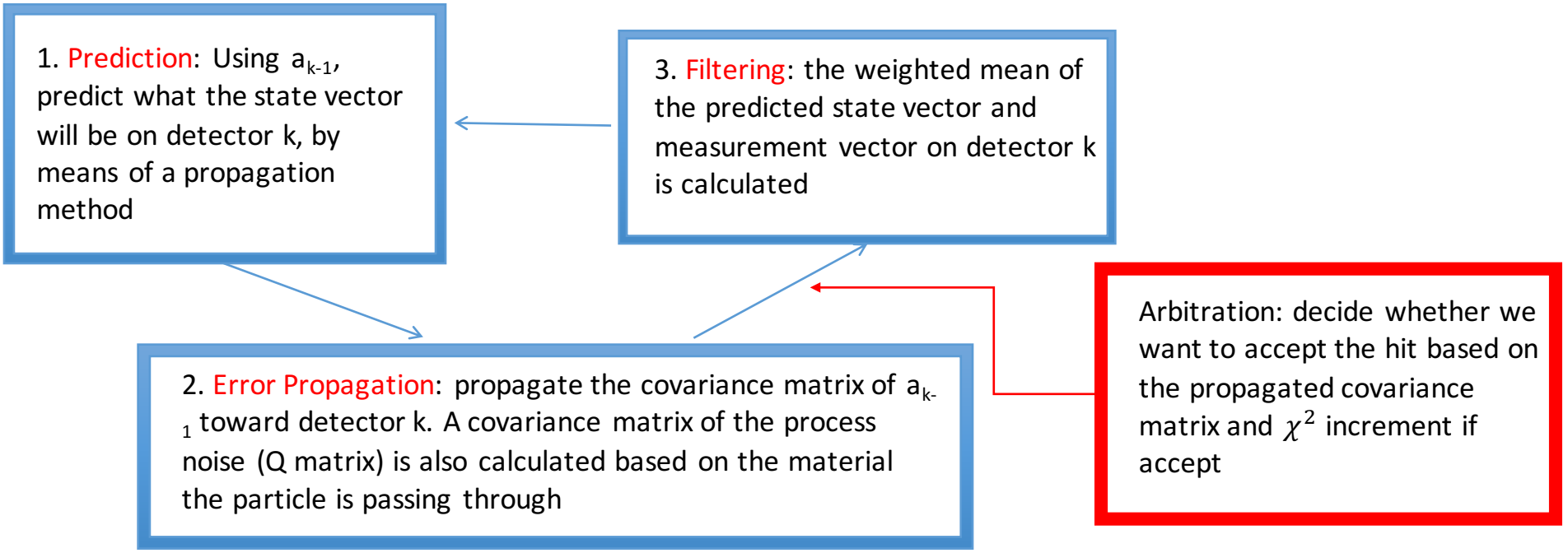
Kalman Filter as the Tracking Finding Algorithm

Kalman Filter: a recursive fitting algorithm based on χ^2 minimization



Kalman Filter as the Tracking Finding Algorithm

Kalman Filter: a recursive fitting algorithm based on χ^2 minimization



Kalman Filter as the Tracking Finding Algorithm

- Using generic state vector to describe the trajectory:
  - $(x, y, t_x, t_y, q/p)$
  - No predefined track model, allow smooth transition between fringe field and strong field
  - Field propagation reply on 4^{th} order Runge-Kutta method (no necessary slow if the step size is large)
  - We need a precise knowledge on the SoLID field map
- Currently the program has treatments for three types of process noises:
  - Coulomb multiple scattering (Molière formula)
  - Ionization (Bethe-Bloch formula)
  - Bremsstrahlung radiation (Bethe-Heitler formula)
- Calculation for the Q matrix and correction for energy are done step by step along the propagation, based on what material the particle is in

Kalman Filter as the Tracking Finding Algorithm -- Seeding

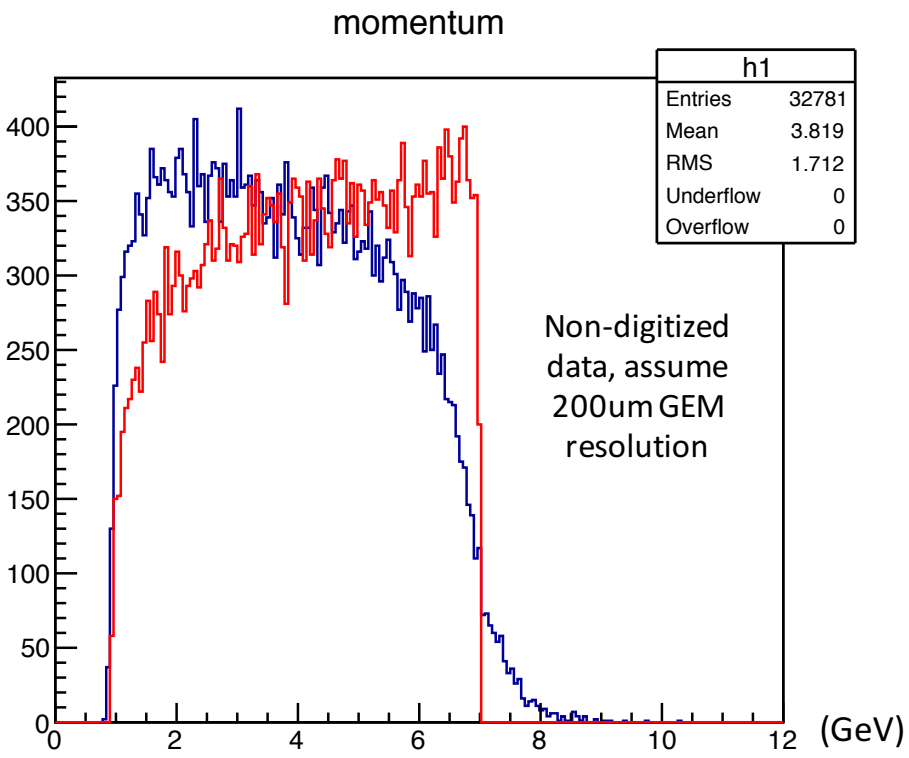
- Seeding: the minimum number of hits that I need to initialize Kalman Filter (3 will be needed in the case of SIDIS FA)
  - Selecting sets of hits from downstream GEM detectors (low hit multiplicty)
  - Quite tricky and time consuming
  - Selection rule cannot be too strict, if the true seed is lost, hard to recover the true track later on
  - Cannot be too lose, large amount of seeds will lead to huge computation time later on
  - Need to consider multiple seed patterns, because our GEMs are not 100% efficient

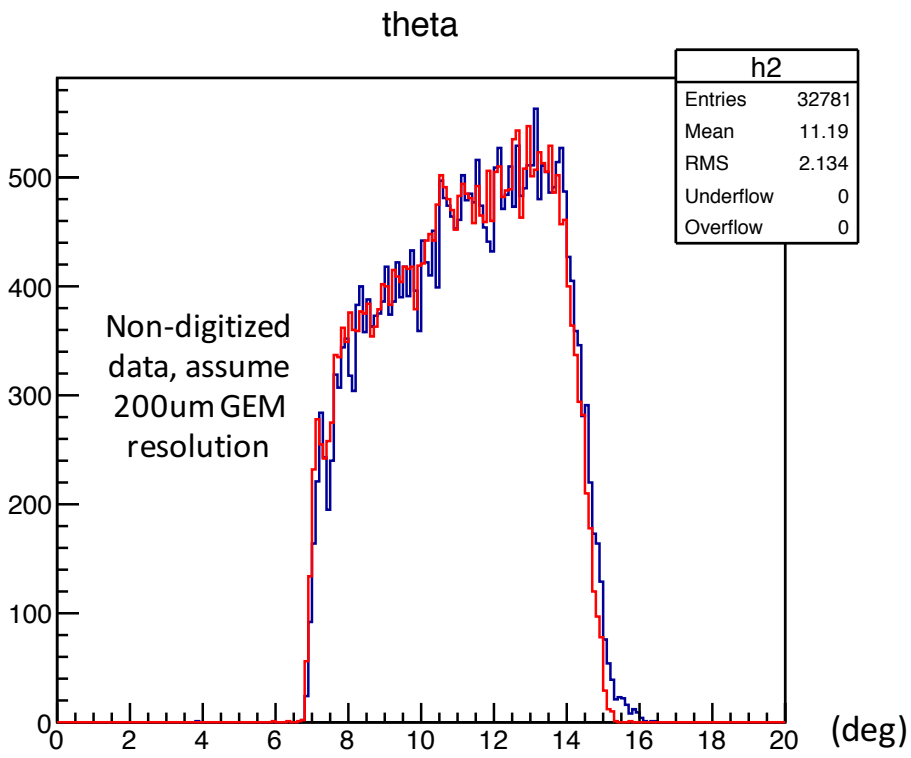
- Currently seeding algorithm largely based on Xin's Progressive tracking, will try to exploit more advanced algorithm later on (some fast global method or machine leaning i.e. boosted decision tree)

Kalman Filter as the Tracking Finding Algorithm -- Seeding

- Current seeding procedure:
  - Have predefined and ranked seed pattern
    - hits from GEM 456, 356, 346 and 345. 2nd GEM is not be considered (4.6k hits on this one)
    - Have predefined maximum seed number (sometimes particles generate small shower and result in huge amount of hits in a small region)
    - Look for seed pattern 456 first (clearly the best one), move on to next seed pattern if the maximum seed number is not exceeded
  - Based on simulation and hint from detectors downstream, zoom in to a region where hits are likely to be (i.e. look at Δr and $\Delta\varphi$ between two hits)
  - With two hits in hand, approximate the track as straight line to look for the next hit (tracks in r-z space are quite straight)
  - With three hits in hand, calculate initial helix, which leads to momentum and angles estimation, and now we can Runge-Kutta propagate the track back to target and FAEC to further examine the track

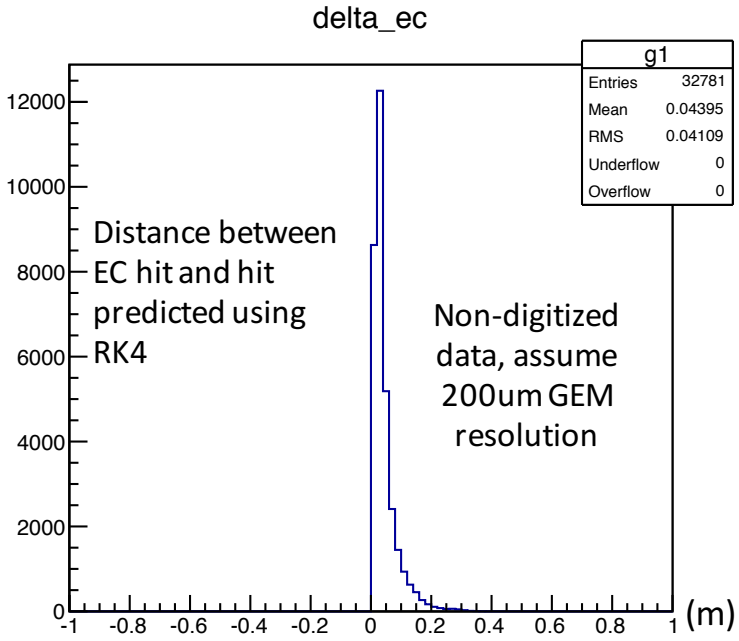
Kalman Filter as the Tracking Finding Algorithm -- Seeding

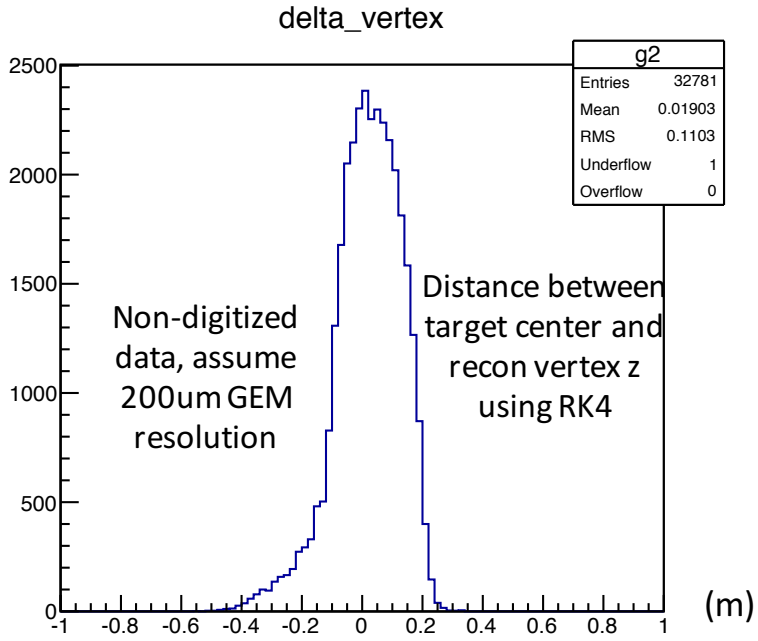




Calculate momentum and theta using helix and hits from GEM 4, 5, and 6. Precision is not required, only sanity cuts on this two variables

Kalman Filter as the Tracking Finding Algorithm -- Seeding





- In the optimal seed pattern (456) takes only ~1 ms to cut total number of combination from ~10^8 down to 29 on average. Take another 1~2ms to finish tracking. (~90% of the final results achieved here)

Kalman Filter as the Tracking Finding Algorithm -- Seeding

- However, consider other seed patterns significantly increase the computation time as we are using GEM plane with higher hit multiplicity. Improvement is small (a few %)

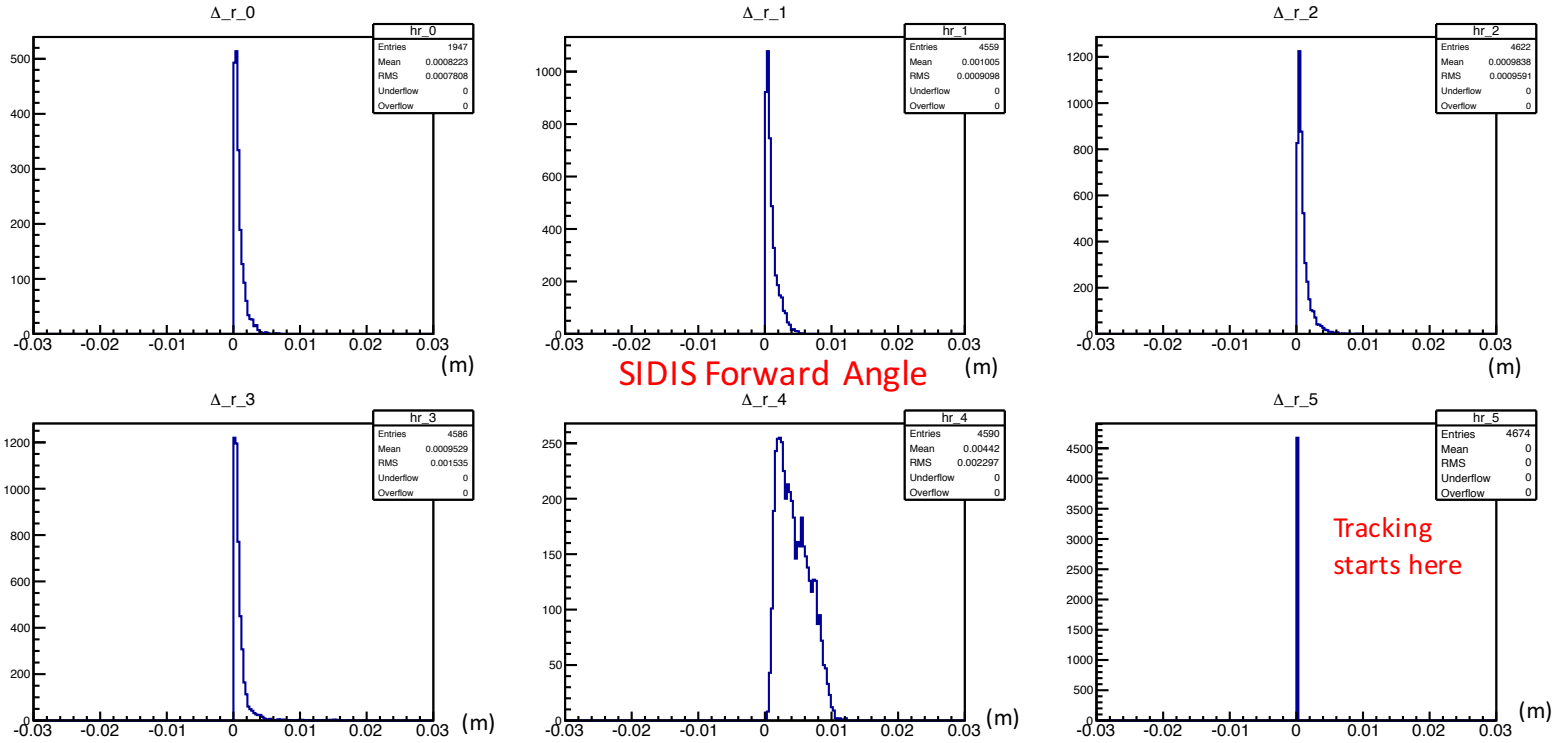
- Lots of repetitive work done here with other seed pattern, i.e. looking partial tracks that have been look before

- Avoid repetition, more efficient seed finding become the bottle neck right now

- Global method such as Hough transformation and Tree Search might be useful and so as boosted decision tree (a type of machine learning method)

Kalman Filter as the Tracking Finding Algorithm – Track Following

- Propagate current filtered state vector to next GEM
- Open a prediction window, if hit found, add and filter it, if not continue
- Can tolerate at most one missing hit

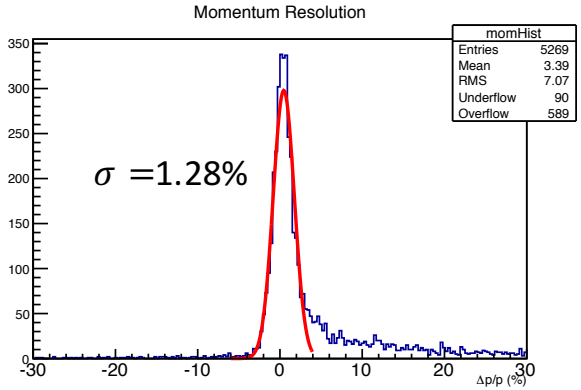


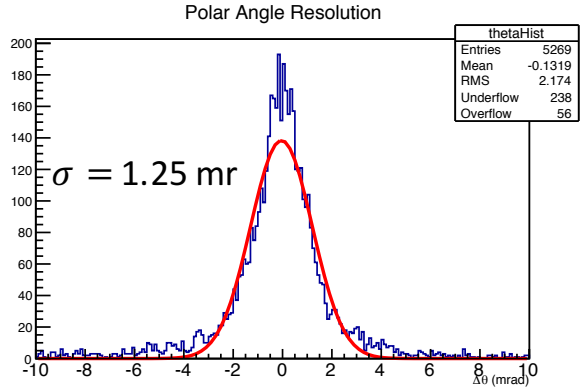
Kalman Filter as the Tracking Finding Algorithm – Track Following

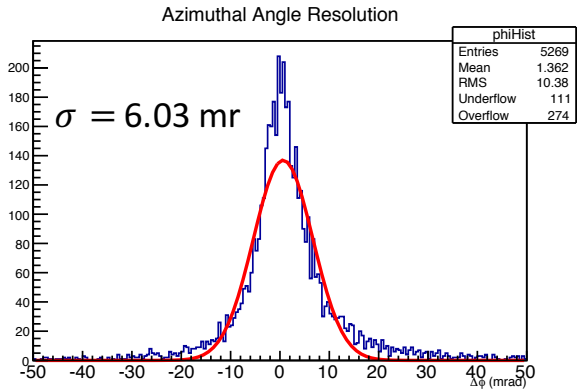
- Many things are not optimized yet:
  - Prediction window should not be a circle but a ellipse based on the characteristic of our GEMs (long axis along radial direction)
  - Covariance matrix is not accurately calculated yet
  - Size of the perdition window is fixed for now, should become auto-adjusted based on the covariance matrix
  - If multiple hits found in a window, adding the closing one to the prediction for now, should split the track later on

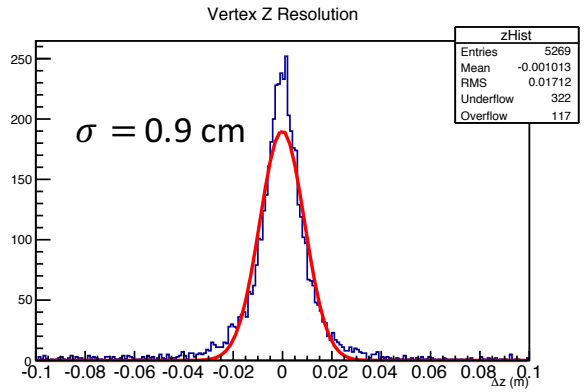
Kalman Filter as the Tracking Finding Algorithm – Final Selection

- Before this step, all tracks stopped at the 1st GEM tracker.
- Since now hit finding is complete, cut on χ^2 per NDF to get rid of some potential bad tracks (<7)
- Propagate the track back to target to add beam spot position and do a finer vertex z cut (<10cm from the edges)





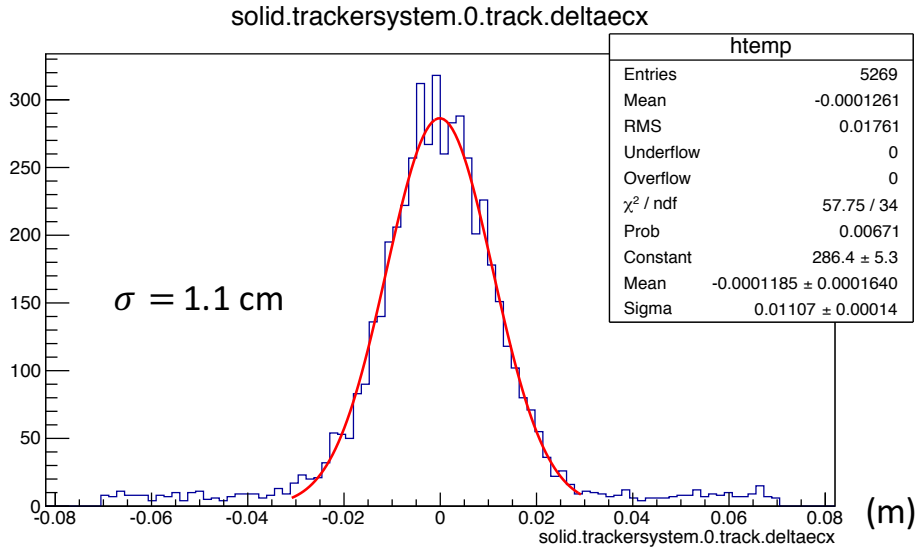


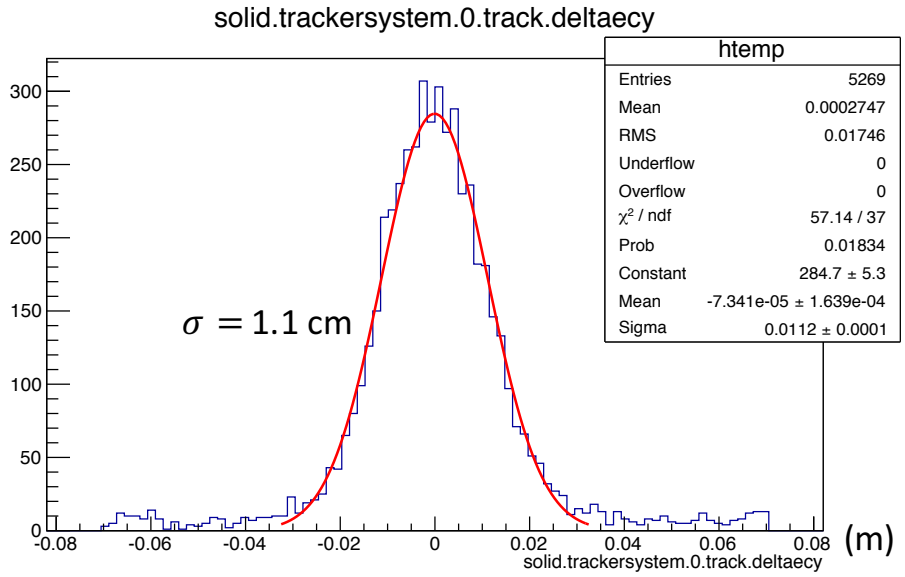


SIDIS Forward Angle

Kalman Filter as the Tracking Finding Algorithm – Final Selection

- Propagate the track back to downstream to do a final match with other detectors
  - Quite a long way back since right now my optimal state vector is at vertex
  - Solution: using Kalman filter smoothing technique, allows reevaluate past site without the need to refit again

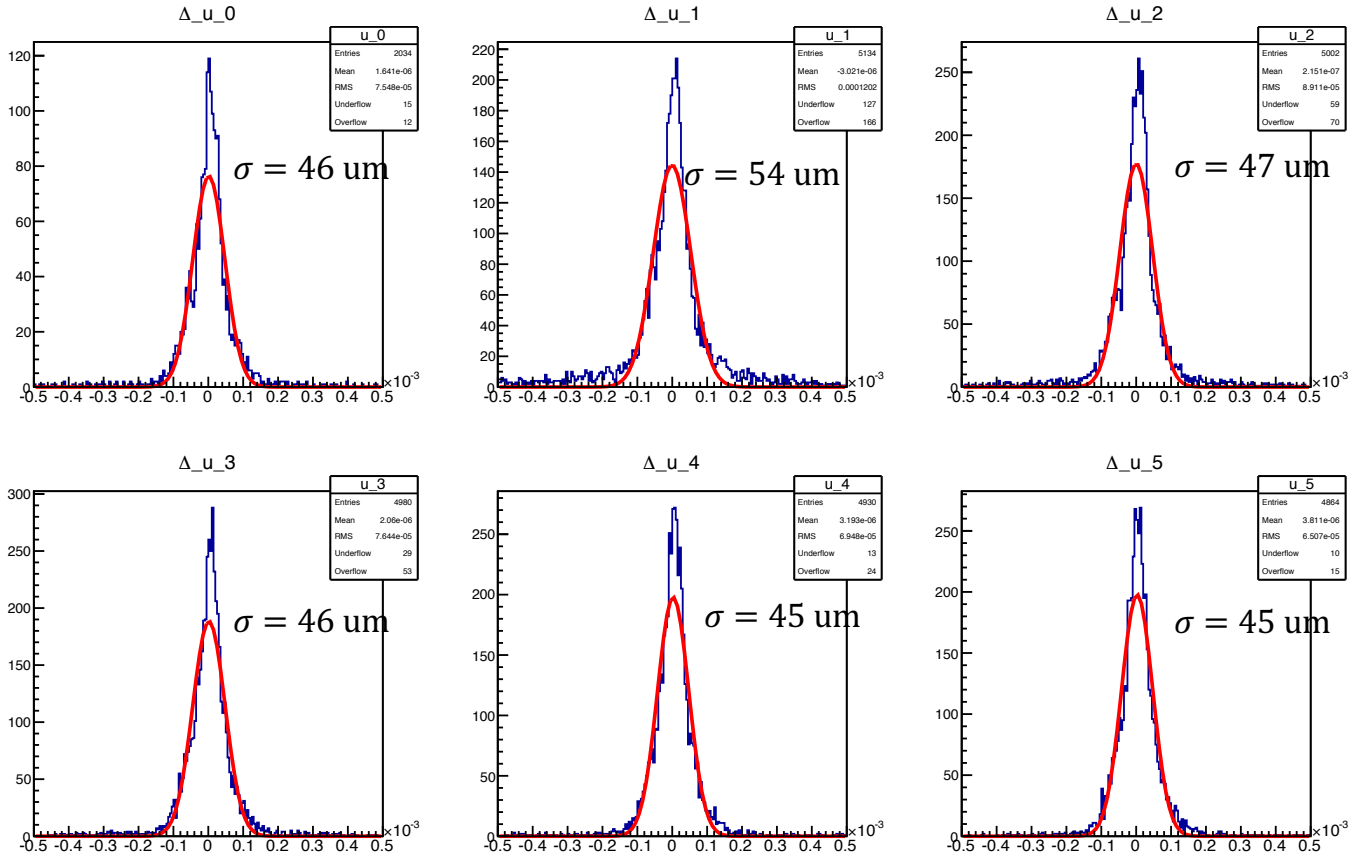




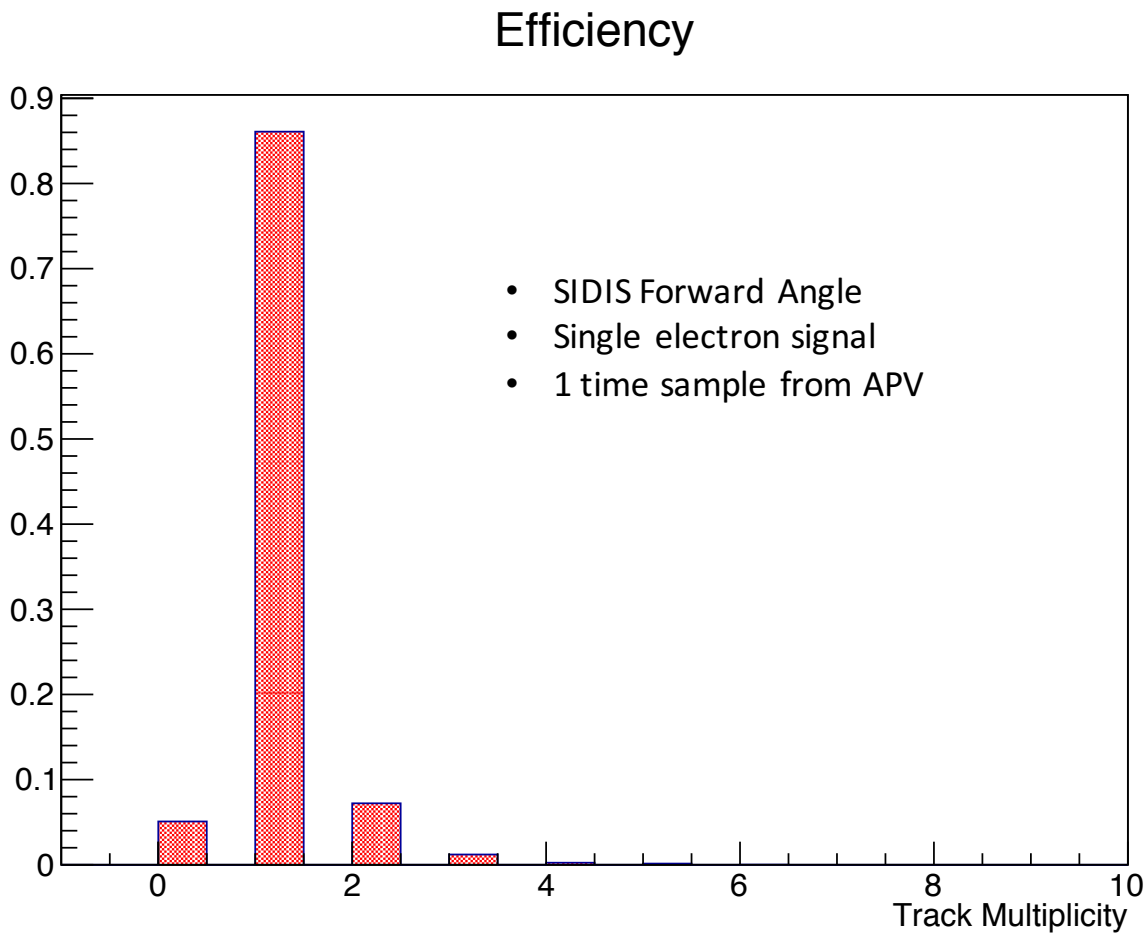
SIDIS Forward Angle

A Few Things about GEM Digitization

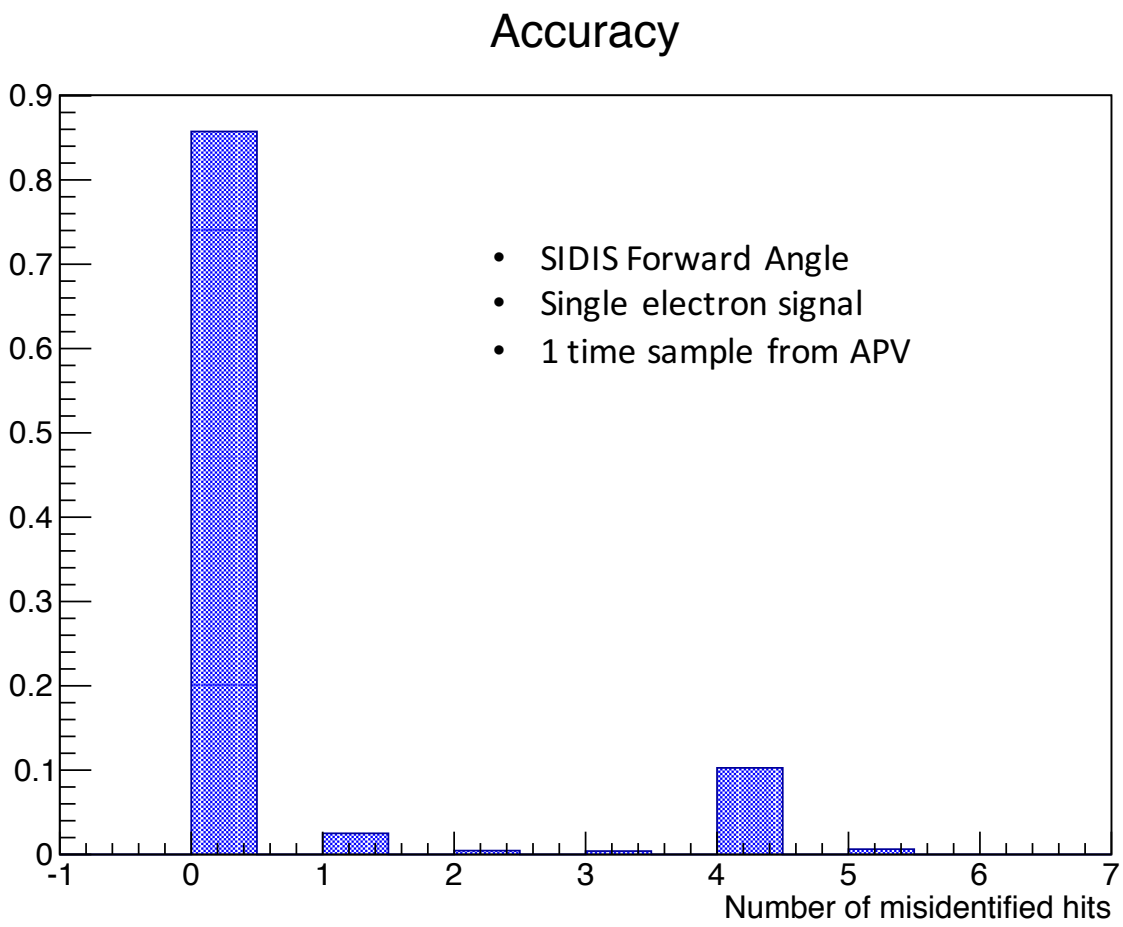
- Adding some electronic noise to the APV signal. $\sigma \approx 20\ ADC$, roughly the noise level to SoLID GEM
- Resolution changes from 30um to ~45um in the case of 0% background. ~54um for the 2nd GEM at 100% background. Still too good to be true
- 100% background



Kalman Filter as the Tracking Finding Algorithm – Efficiency



Kalman Filter as the Tracking Finding Algorithm – Accuracy



Conclusion and What Can Be Improved

- Kalman Filter gives quite promising performance with only one time sample from the APV25
- Timing needs to be improve greatly, I will start from seeding
  - Using only the optimal seed pattern is quite faster (~2.5ms)
  - Consider multiple seed pattern become very slow (over 10ms)
  - Better way of writing C++ code will help also, have not been focusing on this a lot yet
- There are a lot of things we have not considered yet
  - GEM low energy photon response
  - Realistic GEM geometry
  - GEM noise, resolution, trigger jitter…
- But there are lots of useful things I have not exploited also
  - Track length, kinematic constraint…
  - MRPC, Cherenkov and SPD info