New Tracking: Hodoscopes, PU mitigation, and Drell-Yan reconstruction

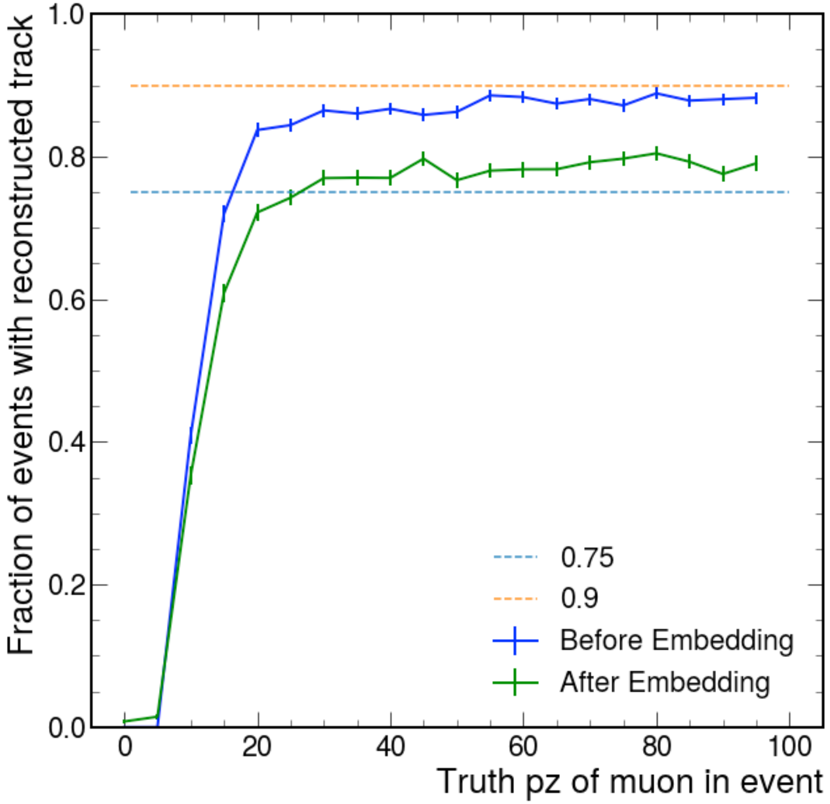
Patrick McCormack, Sebastian Rotella, Duc Hoang, Eric Moreno, Simon Rothman, Noah Paladino, Phil Harris (MIT)

Yongbin Feng, Cristina Mantilla Suarez, Nhan Tran (FNAL)

March 2, 2022

Reminder

- Last meeting: https://seaquest-docdb.fnal.gov/cgi-bin/sso/ShowDocument?docid=9848
- Main content:
  - The dark sectors group has developed a tracking algorithm that
    - Works for **displaced particles** and **prompt particles**
    - Is **more efficient** than the baseline reco algorithm (demonstrated for displaced particles last time)
    - Is **faster** than the baseline algorithm



Reconstruction efficiency for displaced muon gun. Even after embedding, typically processed 5-8 events per second

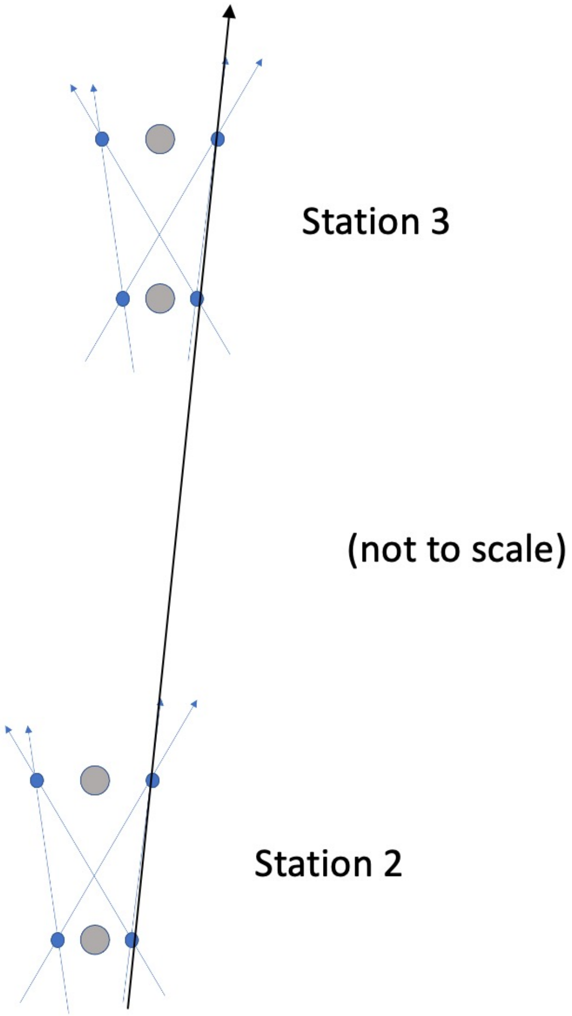
This time

1. Improved pileup mitigation scheme
2. Study in simulated Drell-Yan events
    1. Reconstruction efficiency and timing results
    2. Comparisons with original algorithm

PU mitigation

- Recall from last time the first few steps of the new tracking algorithm:

1. First, we find all **valid hit combinations** in the vertical wires of **station 2**, the left-slanted wires of station 2, and the right-slanted wires of station 2
   - Result: 3 collections of hit combinations
2. We do the same for the three wires slants in **station 3**
   - Result: 3 additional collections of hit combinations
3. We form valid *combinations of the hit combinations* in the vertical wires, left-slanted wires, and right-slanted wires **separately**
   - Matching involves an extrapolation between station 2 and station 3, and there are cuts based on how well the intra-station slopes match and how accurate the extrapolation was
4. Loop over the three collections of st2+st3 hits trying to find valid combinations (if each wire layer has a hit, we'd get 12-hit tracklets)



PU mitigation

- In high-PU events, that last step can be time intensive – if there are 100 combination each of X, U, and V st2+st3 hits, then the loops in step 4 have 1,000,000 iterations, which takes ~10s
- Original basic PU mitigation scheme:
  - Tighten matching requirements for step 3 in high PU events, which results in fewer st2+st3 hit combos, and thus fewer iterations
- Second method:
  - Bin st2+st3 hit combos by position in station 2, and only loop over “nearby” combinations in X, U, and V
  - Minor problem here is that X, U, and V are treated as separate coordinate systems, so e.g. the X and U positions of a particle can be somewhat significantly different
- Note: after last time’s presentation I ran on a DY sample and was slightly disappointed by the efficiency.
  - Events were failing step 3 from last slide, so I loosened my matching requirements – this increases reco time!

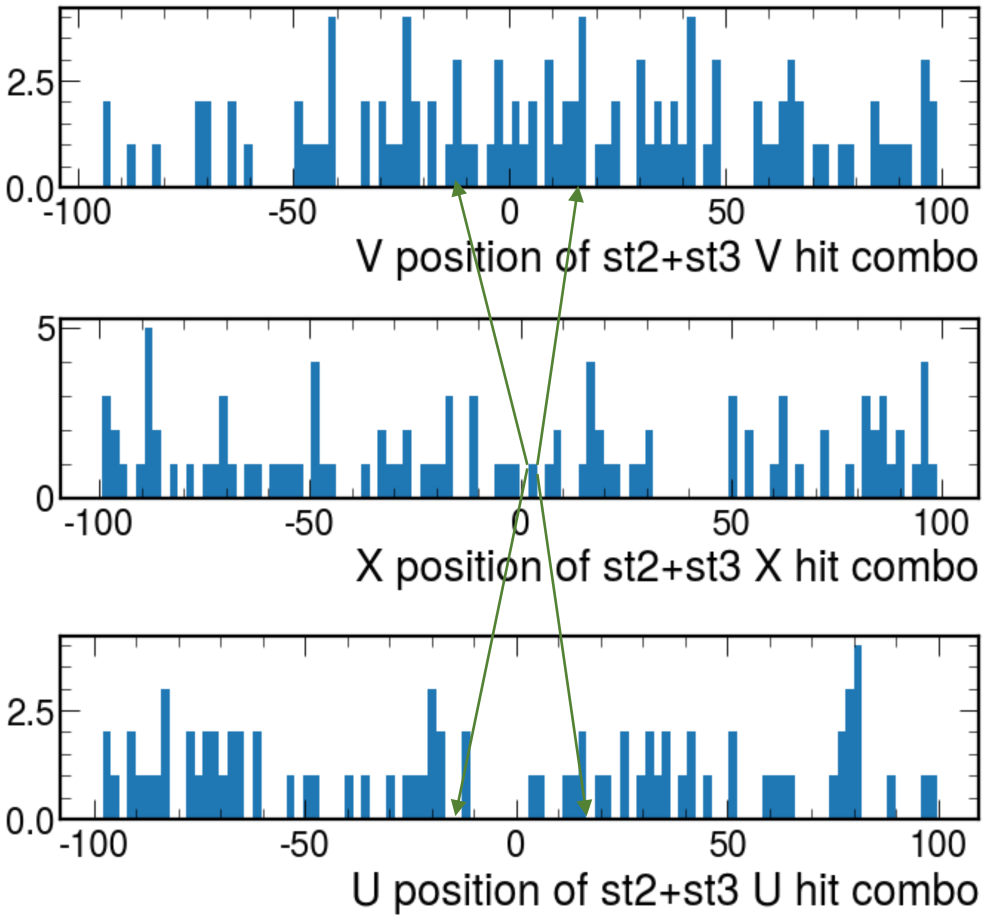


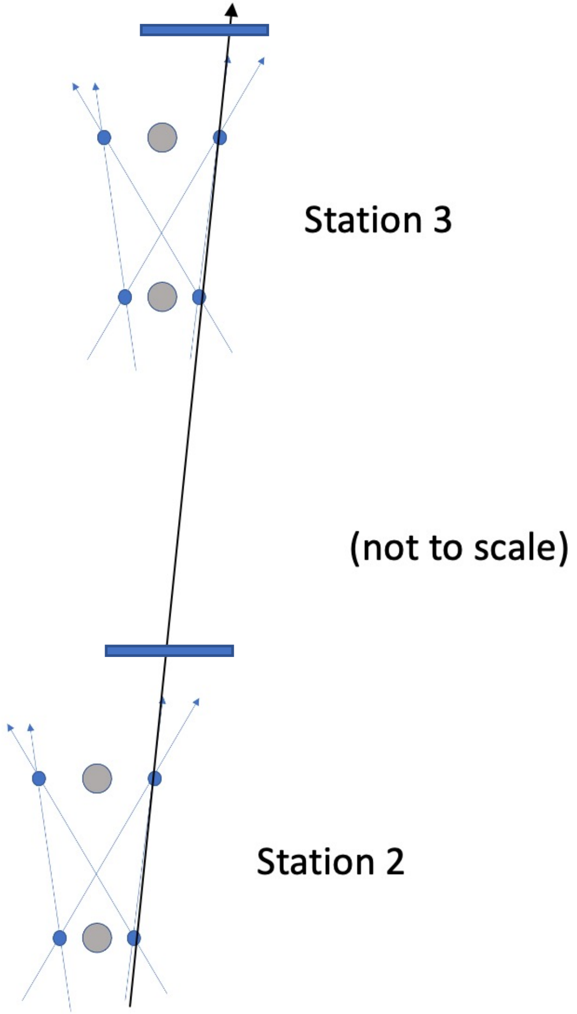
Illustration of bin-based PU mitigation scheme. This example event has 100 X st2+st3 combos, 90 U combos and 110 V combos. Without any PU mitigation, each X combo will be compared to every X and V combo, but with mitigation, we only compare to “nearby” U and V combos, here indicated by the green boundaries for a single X combo

PU mitigation - Redux

- One piece of information that I had been neglecting up until now is the hodoscope hits
- When combining st2+st3 hits (step 3 from slide 4), I can check whether the trajectory that would be needed to connect the hits passes through station 2 and station 3 hodoscopes
    - This is very accurate out of the box for X wires, and we can use larger windows for U and V trajectories
- New PU mitigation scheme: rather than binning by position in station 2, we can basically bin by which station 2 hodoscopes the tracklet can match with *and* which station 3 hodoscopes the tracklet can match with
    - In a sense, this is binning by both position and slope

PU mitigation - Redux

- Practically speaking how this works is the following:
  1. I build all acceptable st2+st3 X hit combinations. Each combination keeps track of the st2 and st3 hodoscopes that it could have passed through (as there are occasionally more than one)

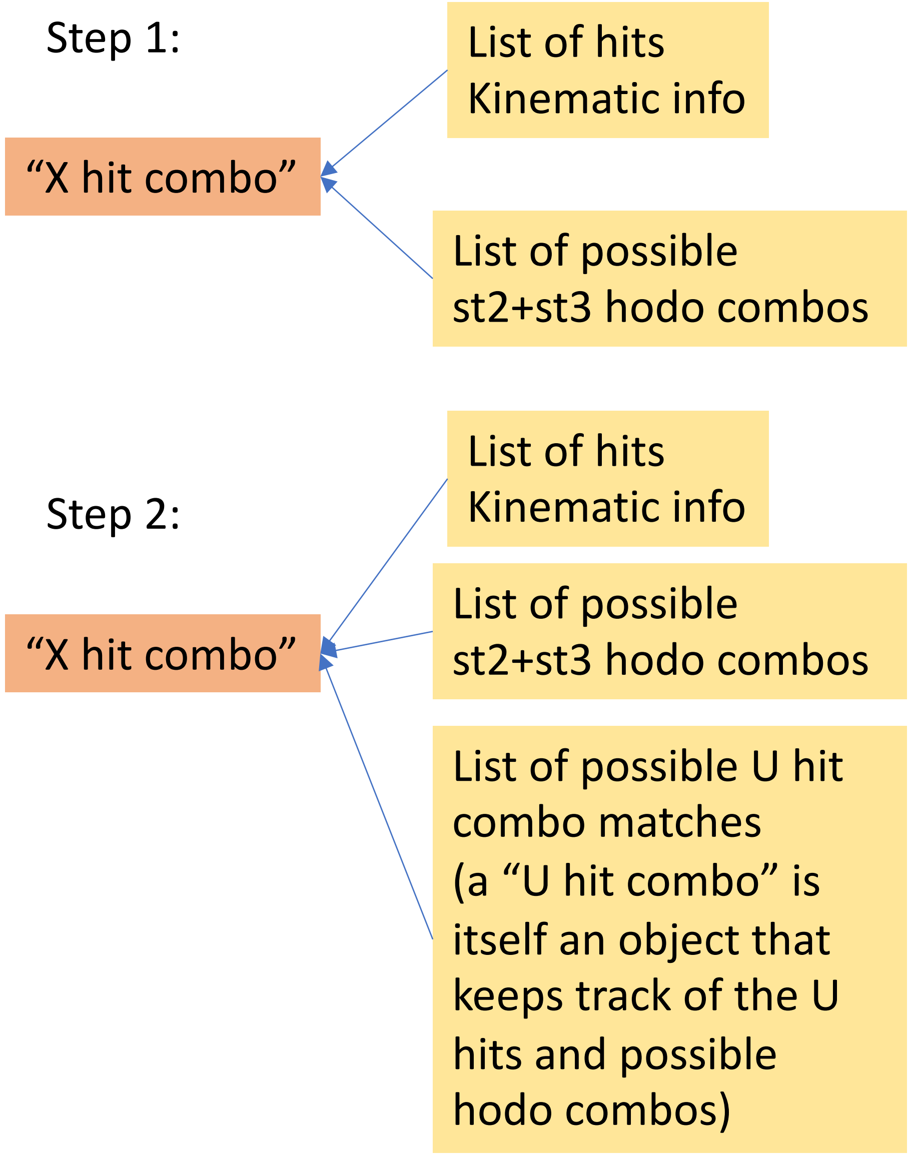


Each st2+st3 X hit combo keeps track of possible station 2 + station 3 hodoscope hit combinations (this example only has 1 hodo combo)

PU mitigation - Redux

- Practically speaking how this works is the following:
  1. I build all acceptable st2+st3 X hit combinations. Each combination keeps track of the st2 and st3 hodoscopes that it could have passed through (as there are occasionally more than one)
  2. I build all acceptable st2+st3 U hit combos. For each I also find the possible st2 and st3 hodoscope combos. For each U hit combo, I loop over all the X hit combos, finding each that could have the same hodo combo. I keep track of every match

What's in each st2+st3 X hit combo object?



Step 1:

"X hit combo"

List of hits
Kinematic info

List of possible st2+st3 hodo combos

Step 2:

"X hit combo"

List of hits
Kinematic info

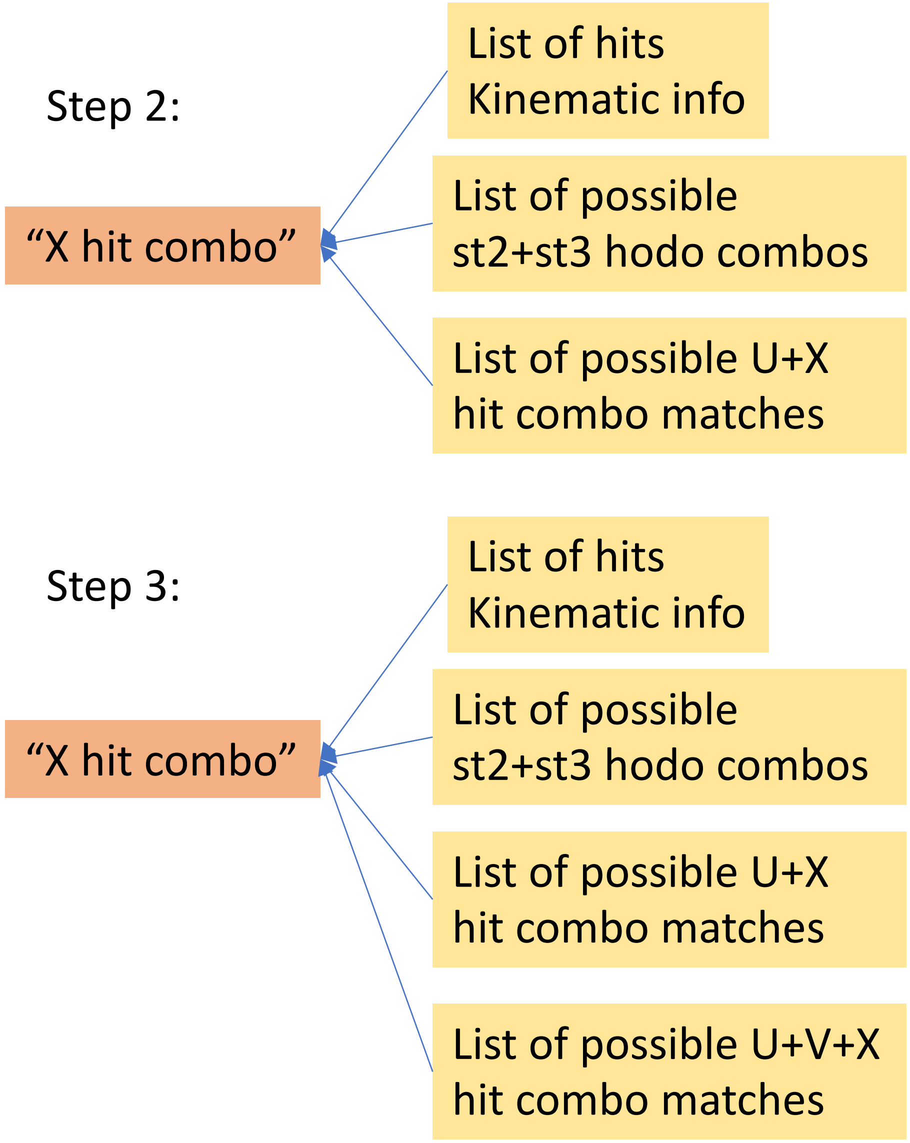
List of possible st2+st3 hodo combos

List of possible U hit combo matches (a "U hit combo" is itself an object that keeps track of the U hits and possible hodo combos)

PU mitigation - Redux

- Practically speaking how this works is the following:
  1. I build all acceptable st2+st3 X hit combinations
  2. Find possible U matches for each X combo based on hodo matches
  3. I build all acceptable st2+st3 V hit combos and find hodoscope matches. For each V combo, I loop over the possible U+X combos, finding ones that share hodoscope matches
  4. In the end, each st2+st3 X combo has a full list of possible associated U and V st2+st3 hit combos based on hodoscope matching

What's in each st2+st3 X hit combo object?



Step 2:

"X hit combo"

List of hits
Kinematic info

List of possible st2+st3 hodo combos

List of possible U+X hit combo matches

Step 3:

"X hit combo"

List of hits
Kinematic info

List of possible st2+st3 hodo combos

List of possible U+X hit combo matches

List of possible U+V+X hit combo matches

PU mitigation - Redux

- In this PU mitigation scheme, step 4 from slide 4 is now just a double loop over all st2+st3 X hit combos and their U+V combos
- I also have implemented dynamic controls over the number of possible combinations by potentially tightening matching requirements depending on the number of combinations as the reco chain progresses

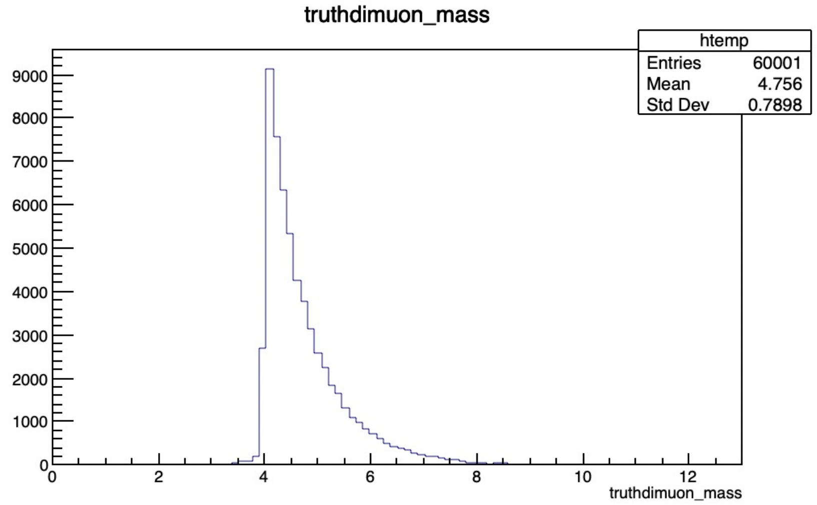
- Near future: X+U and X+U+V combo matching can be improved by using measurements of positions and slopes

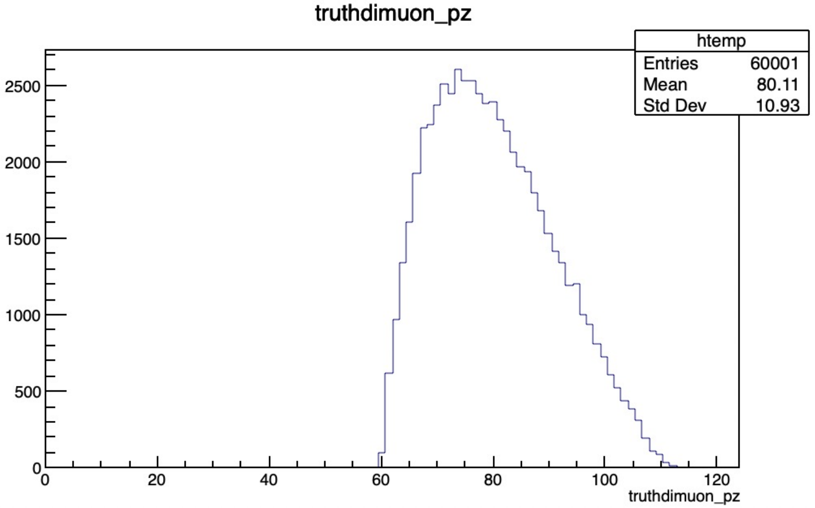
This time

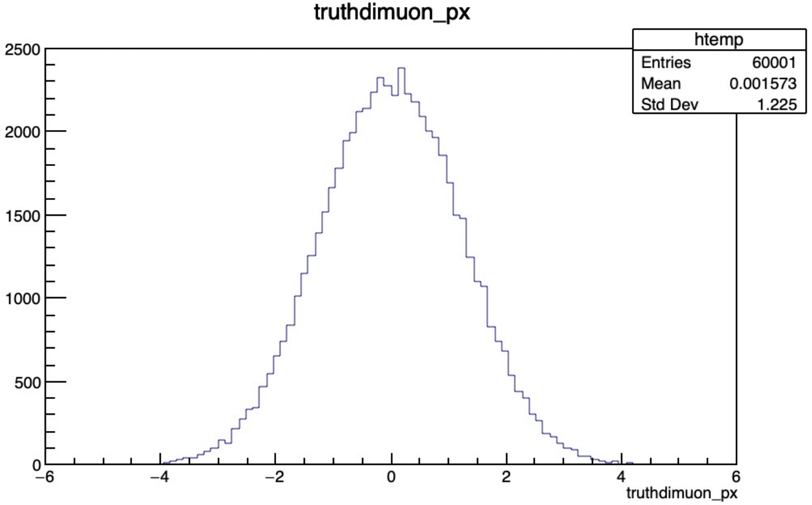
1. Improved pileup mitigation scheme
2. Study in simulated Drell-Yan events
    1. Reconstruction efficiency and timing results
    2. Comparisons with original algorithm

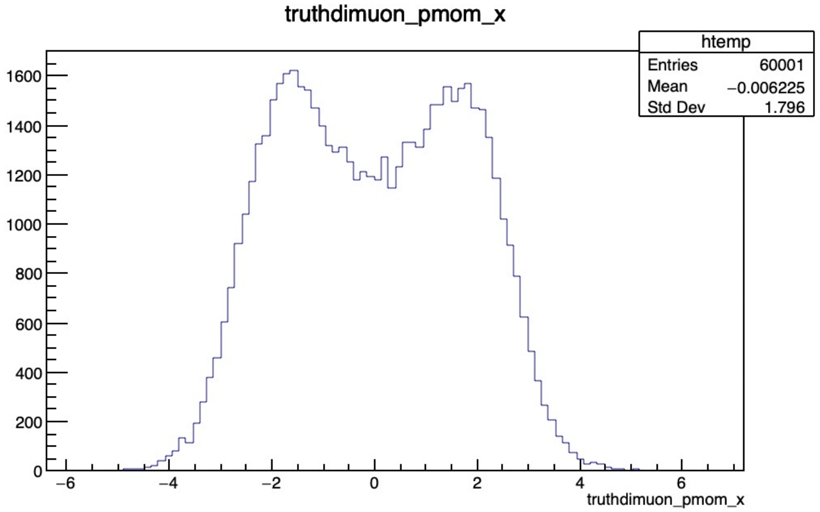
Now something that's easier to understand

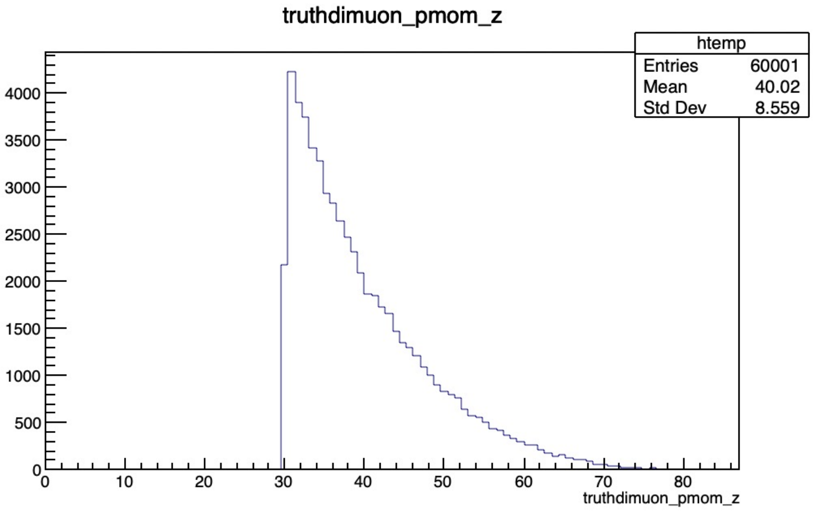
- I generated 30,000 Drell-Yan mumu events with the following kinematic info











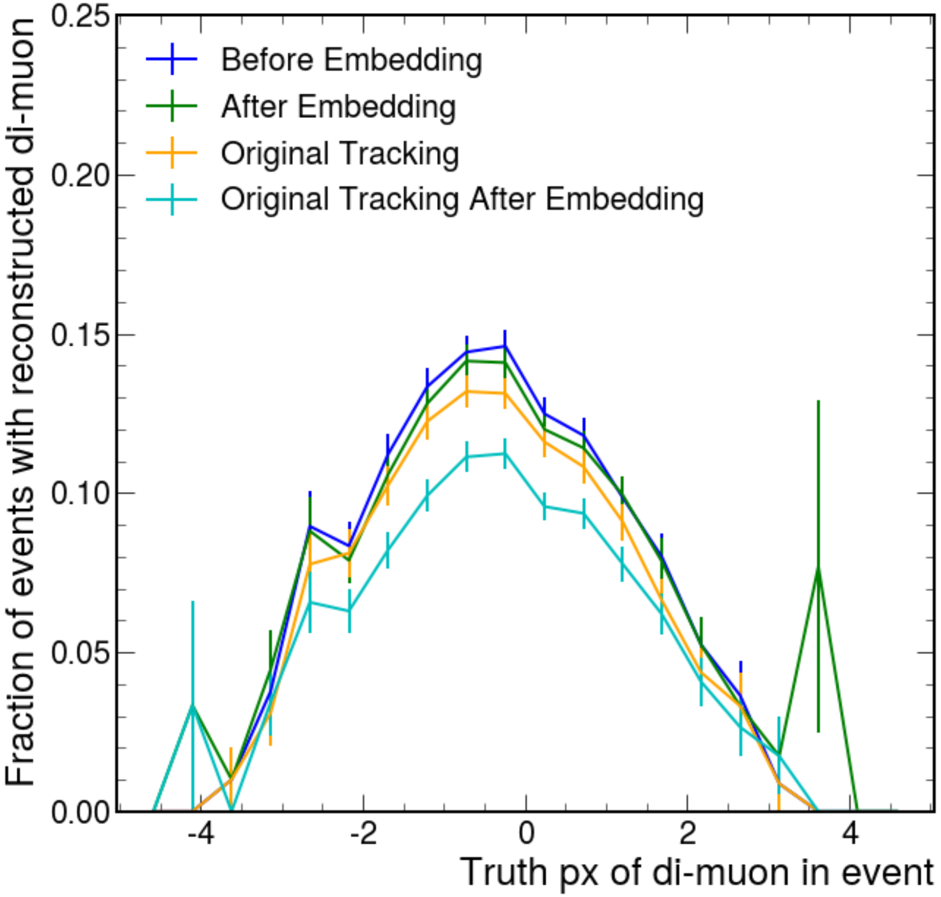
- Dimuons generated at (0,0,0) cm
- For each event I applied DC efficiency and resolution emulation (de-randomized for each event for reproducibility)

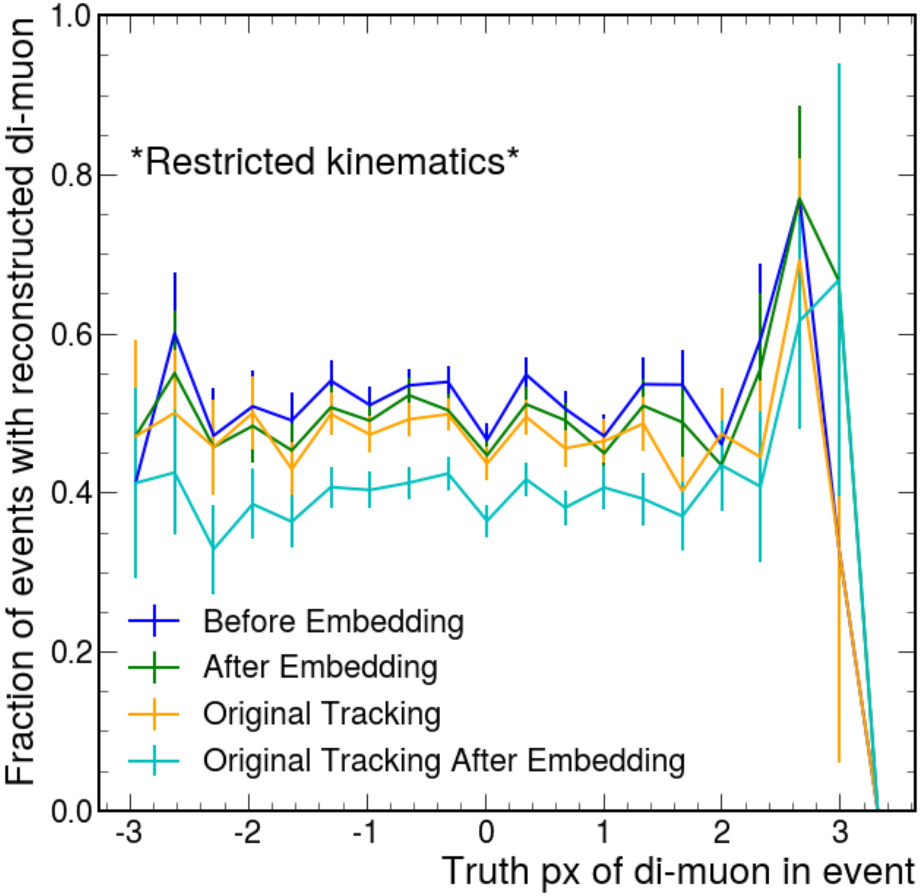
Drell-Yan performance

- I compared 4 different reconstruction scenarios:
  1. New tracking algorithm without data overlay
     - (Uses PU mitigation scheme described in preceding slides, but without PU, this doesn’t do anything)
  2. New tracking algorithm with data overlay
     - Using PU mitigation scheme described in preceding slides
  3. Original tracking algorithm without data overlay
     - Code modified slightly to perform correct hodomask comparisons
  4. Original tracking algorithm with data overlay
     - Uses a cut of 300 hits in either D0, D2, D3m, or D3p for PU mitigation
     - Code modified slightly to not consider D1 hits at all

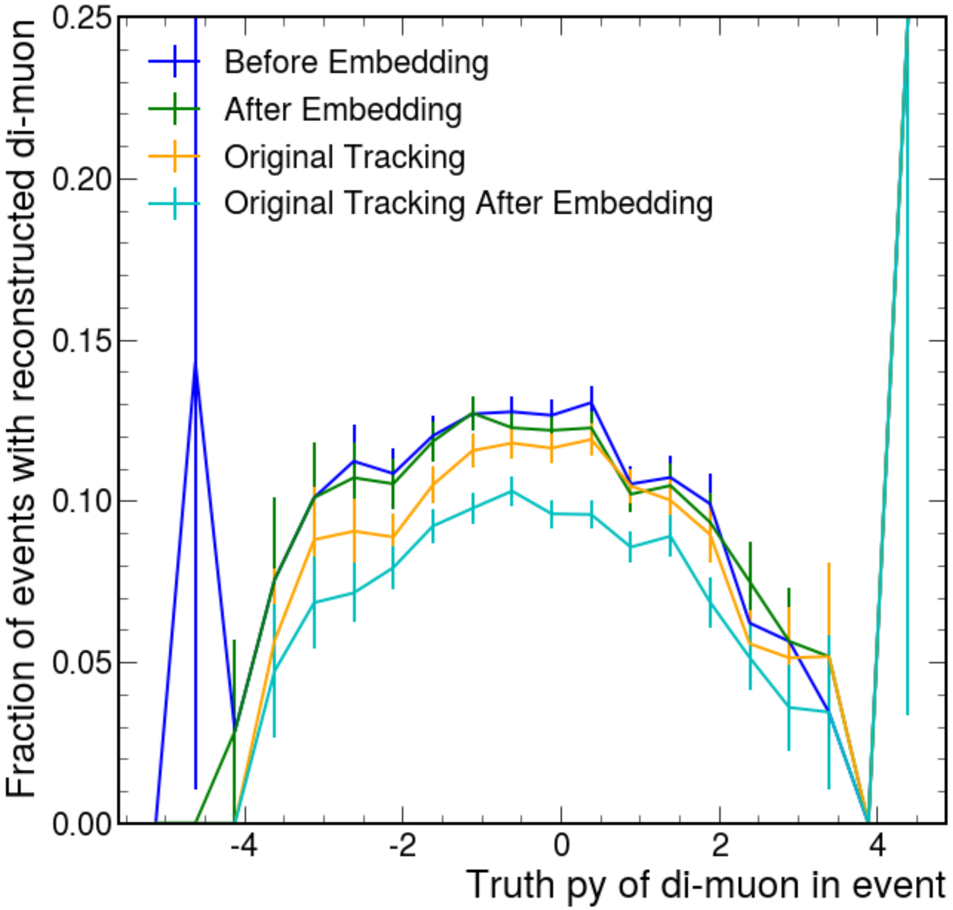
Drell-Yan performance

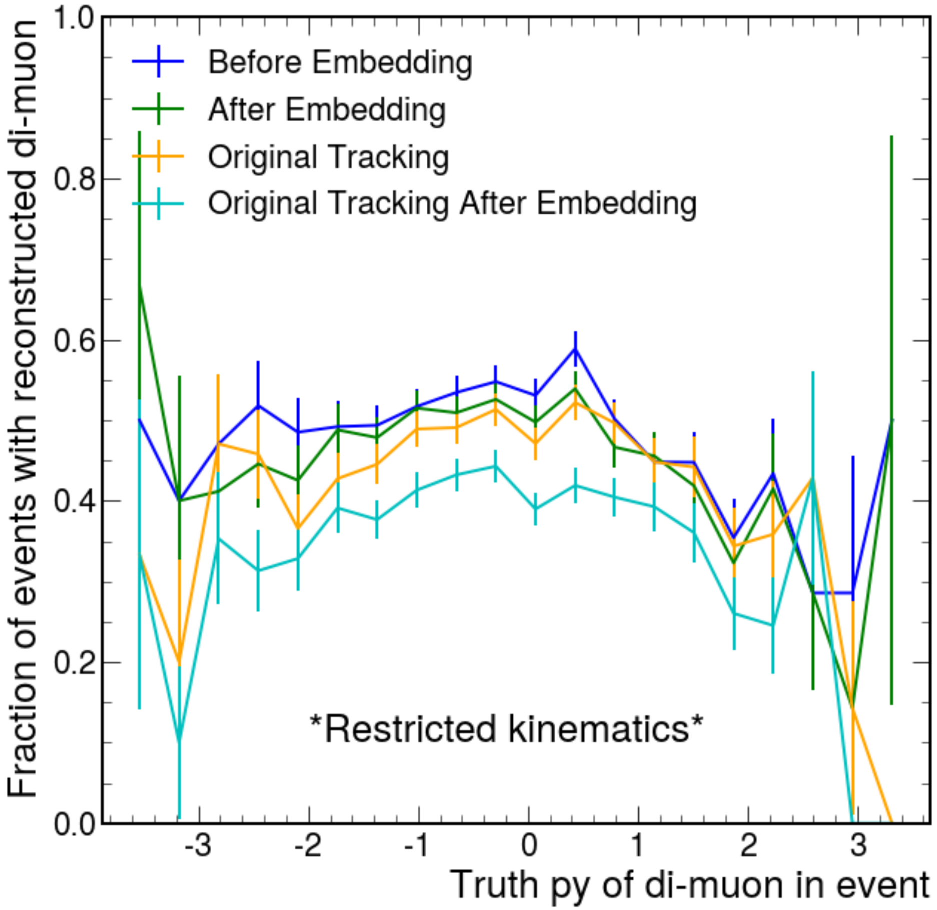
In this and following slides, “restricted kinematics” means that I required that both DY muons in the event have an |x| and |y| value in station 3 less than 100 cm



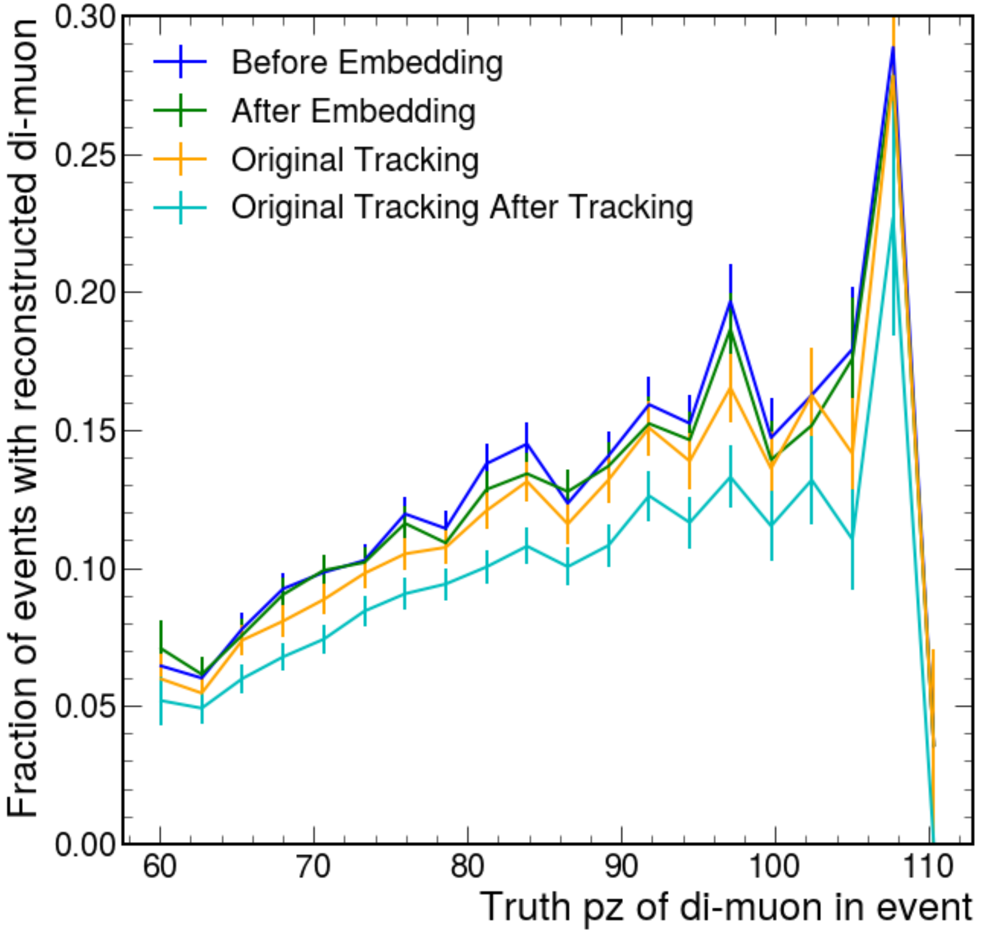


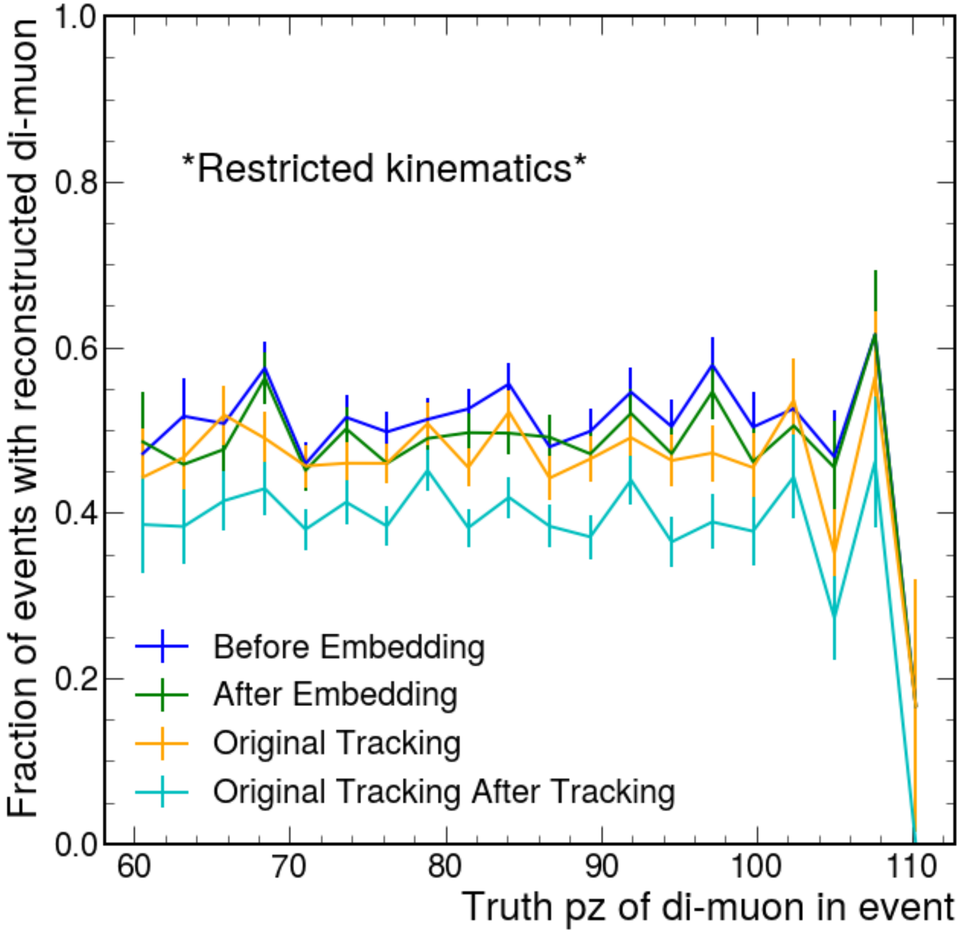
Drell-Yan performance



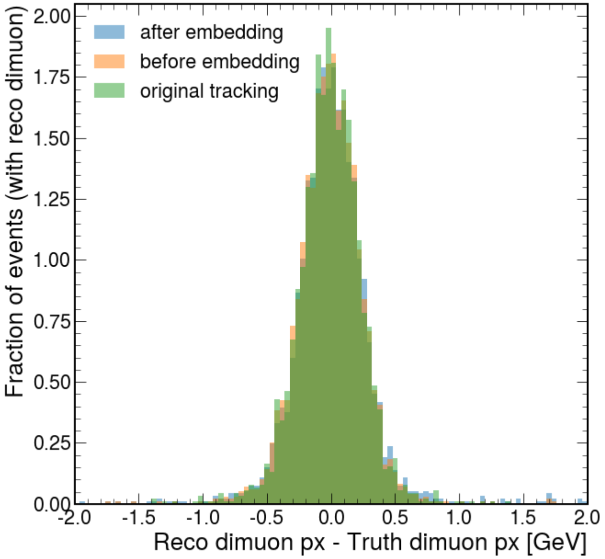


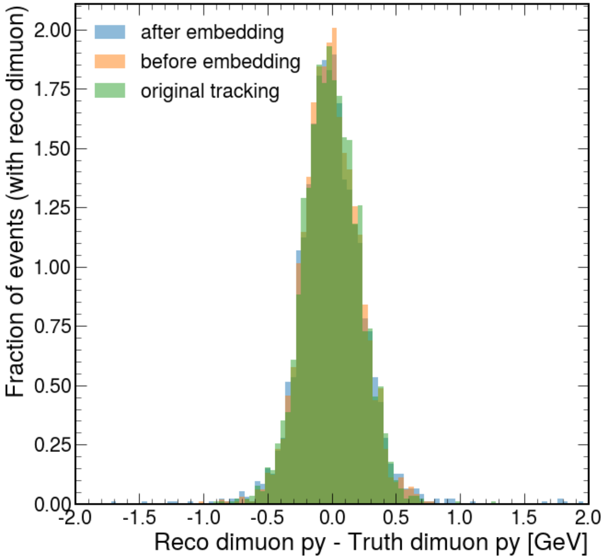
Drell-Yan performance

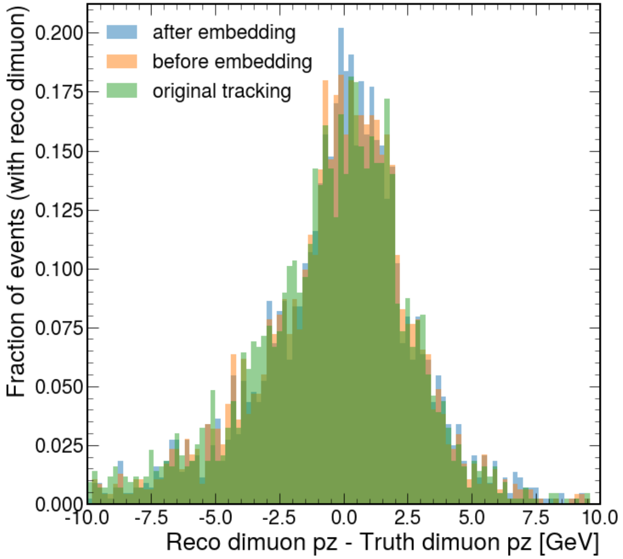


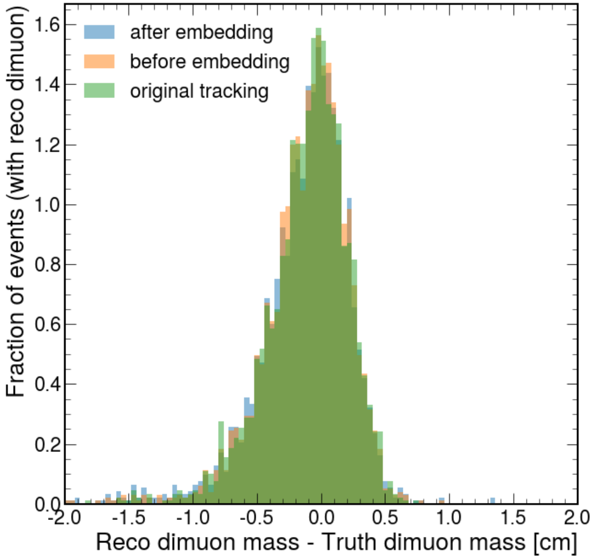


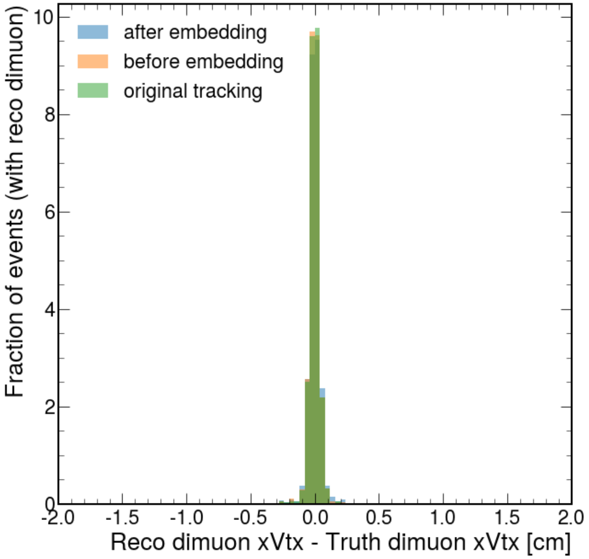
Drell-Yan performance

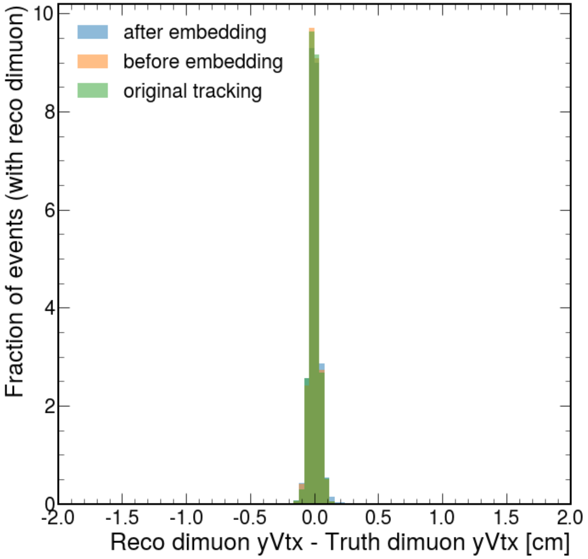


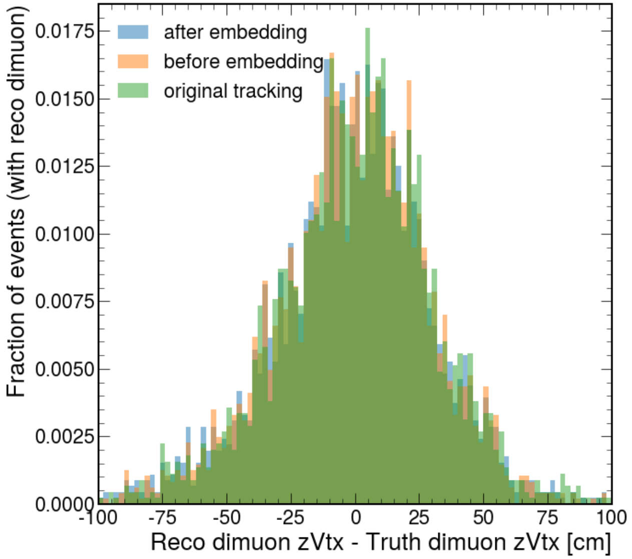






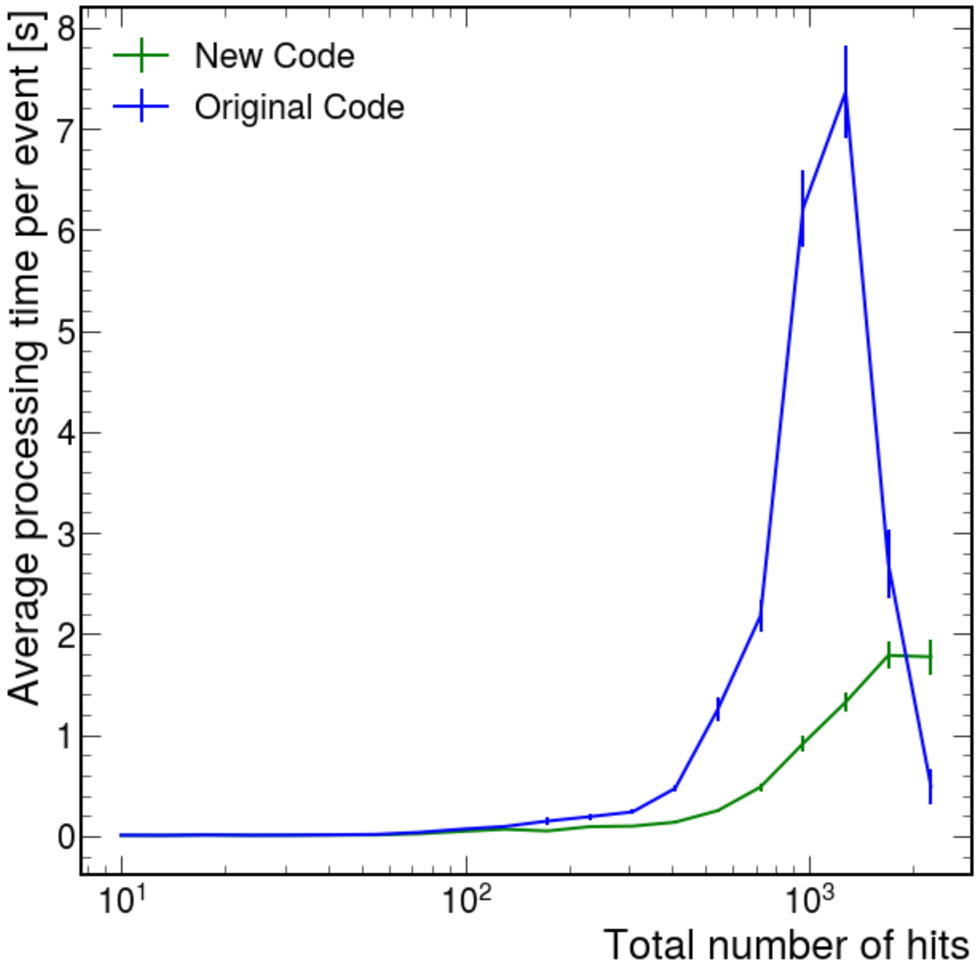




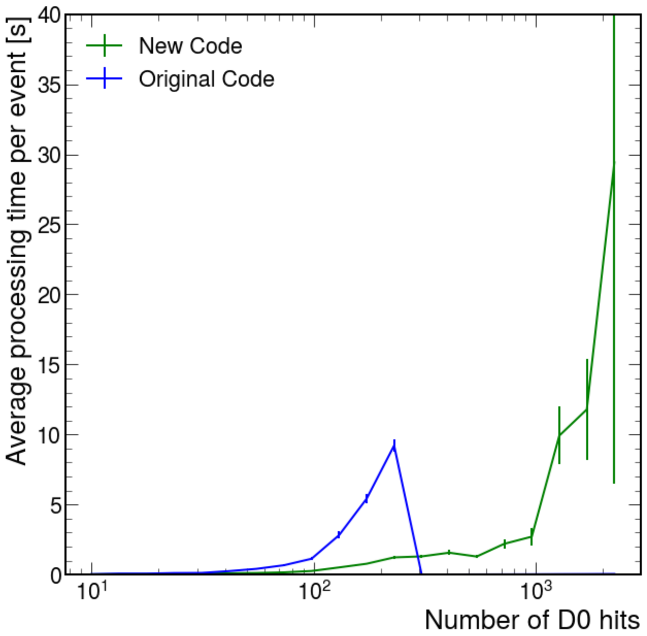


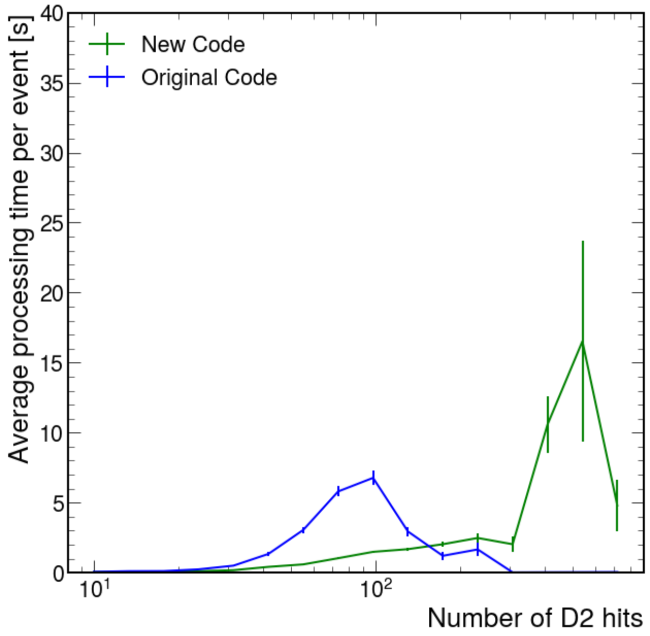
Di-muon resolution seems largely unchanged when comparing old/new algorithms

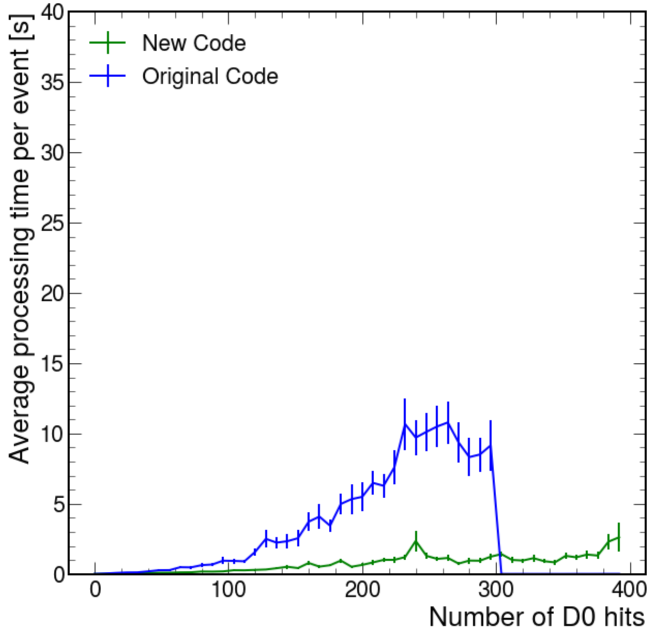
Drell-Yan performance

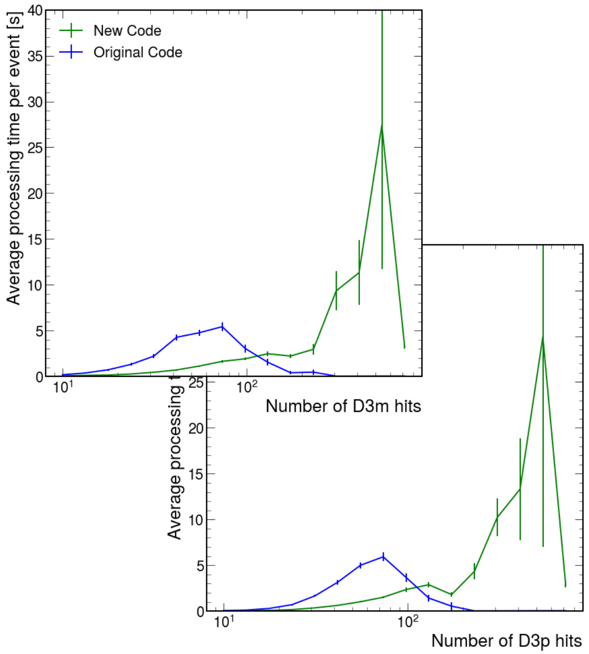


Processing the 30k data-embedded DY events took twice as long overall with the original code. This is despite that the new code processes the highest-PU events, while the original code *does not*.









Some remarks

- With the new tracking algorithm, you get
  - ~25% relative increase in DY reconstruction efficiency (after data embedding) ☺
  - 50% reduction in processing time and less loss due to high-PU events ☺
  - Ability to reconstruct prompt and displaced tracks ☺
- Kind of a win-win-win
- Extremely messy, still-in-development branch of code:
  - https://github.com/wpmccormack/e1039-core/tree/patrick_new_tracking_newerHodo/
  - Main algorithm is here: https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_newerHodo/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx

Future plans

- Improve **low-pz muon reconstruction** efficiency (current reconstruction extends down to pz ~10 GeV)
  - Important for Dark Photon and J/Psi events
  - This should be ~straightforward, though might have negative impact on timing
- Improve resolution/**track quality**
  - I need to add in a little track cleaning and double checks on hit sign assignment
- **Fake rate** studies
  - Goes hand-in-hand with quality improvements
- Clean up code!!
- Further future:
  - Run on more recent analysis data (per Kun's comments last time)
  - GPU-enabled tracking/ taking advantage of parallelization?
  - Re-analysis of old data?