**Implementing Displaced Particle ID**

**Step 0: Background reading**

* Aram might have already pointed you to this document, but this is a paper describing DarkQuest: <https://arxiv.org/pdf/2203.08322.pdf>
* And here is a detailed description of the SpinQuest experiment + detector: <https://arxiv.org/pdf/1706.09990.pdf>

**Step 1: Getting Simulation and Reconstruction Code Set Up**

* The very first thing that we’ll want to do for this project is to get some particle gun simulations up and running. By particle gun, I mean that we can *simulate* particles with with energy, momentum, and flavor of our choosing from any position we want
* To get the reconstruction code set up, please source this script: “[DQ\_install\_script](https://docs.google.com/document/d/1-IeDxgUu5Tk8I92fH4ZQzc9f81YFC8OHYRKHL6AHFFI/edit?usp=sharing).sh”
  + I mean, you’ll have to copy the text in that google doc into a script on the SpinQuest machine, then you can do e.g. “source <scriptname>.sh”
  + Within the script, there are two additional files that are referenced. Could you copy those into separate scripts and then reference them in the appropriate places in the main script? E.g., if you copy the “[setup\_mye1039\_tmp](https://docs.google.com/document/d/17BzkHdsZnpalmJeuvAwTM9mKGvBEuGdcvTdyXUMhbCk/edit?usp=sharing).sh” google doc into a file called “setup\_mye1039\_tmp.sh”, then in the “DQ\_install\_script.sh” script, the line that references to the setup\_mye1039\_tmp.sh script should be
    - sed "s@XXX@$PWD@g" setup\_mye1039\_tmp.sh > setup\_mye1039.sh
* I’m hoping that the setup script works out of the box, but it may not. Multiple people have run it successfully on the MIT computing resource that we normally use, but the spinquest machine might not be set up in exactly the same way. If the script runs into problems, let me know
* Also, note that compilation may take ~10-20 minutes or so, so don’t be too surprised if it takes a while

**Step 1.1: IMPORTANT OPERATIONAL NOTES**

* Whenever you log out and log back in, you need to navigate to <top directory where you set up the code>/e1039-core/DarkQuest/e1039-analysis/SimHits. Then do
  + source setup\_mye1039.sh
* If you change anything in the reco code, which will mostly be the code in <top directory where you set up the code>/e1039-core/packages/reco/ktracker, then you need to navigate to <top directory where you set up the code>/e1039-core/ and compile with e.g.
  + ./build.sh -s packages/reco/ktracker
* If you change anything in the ntuple-making code, which is in <top directory where you set up the code>/e1039-core/DarkQuest/e1039-analysis/SimHits/src, then you need to navigate to <top directory where you set up the code>/e1039-core/DarkQuest/e1039-analysis/SimHits/work and do
  + make
  + make install

**Step 2: Generating Some Particle Gun Events**

* Ok, so now hopefully the code is set up (and if you logged back in, then you sourced the setup\_mye1039.sh script referenced above)
* Then navigate to <top directory where you set up the code>/e1039-core/DarkQuest/e1039-analysis/SimHits/macro
* You can either run the following command to generate a “simulation” command:
  + python run\_sim.py -n 50 --sim 'gun' --gun 'positron' --zvertex 520 --displaced
    - -n indicates the number of events
    - --sim indicates the type of simulation
    - --gun indicates the particle type
    - --zvertex indicates the z-position of the particle
    - --displaced indicates that the particle does not originate from the the main SpinQuest target
* Or you can just run simulation and reco directly with something like:
  + root -b -q RecoE1039Sim.C\(50,3,3,520.00,true,true,false,\"\",\"\",\"output.root\",\"./\",\"/pnfs/e1039/persistent/users/apun/bkg\_study/e1039pythiaGen\_26Oct21/10\_bkge1039\_pythia\_wshielding\_100M.root\",0\)
    - There, the first input is the number of events
    - The second input specifies the simulation type
    - The third input specifies the particle type
    - The fourth input specifies the location where the particle is generate
    - The fifth input specifies that the particle is “displaced” (ie not produced at the SQ target)
    - The sixth input specifies that an ntuple should be created
    - The seventh input tells the code whether or not to run pileup
  + You can see what the various options are here: <https://github.com/wpmccormack/DarkQuest/blob/addTimer/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L24-L62>
    - Note that this input specifies the name of the output ntuple: <https://github.com/wpmccormack/DarkQuest/blob/addTimer/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L58>
  + Also note that you can change information about the particle gun simulation here: <https://github.com/wpmccormack/DarkQuest/blob/addTimer/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L258-L263>
    - set\_vertex\_distribution\_mean tells you where the particle gun is centered at
    - set\_vertex\_distribution\_width specifies the ranges where the particle gun extends around the center
    - set\_pxpypz\_range allows you to specify the momenta ranges for particles generated by the particle gun

**CHECKPOINT 1:**

* To summarize this first round of instructions
  + Could you set up the code as specified above?
* Then could you generate 100 muon gun events with a z vertex location of 520 cm? Once you’ve done that you should have an output ntuple, which is a “.root” file. By default, the file’s name is “output.root”
  + You can open up that file by doing: root -l output.root
  + Then in the prompt, you can type in “new TBrowser”, which opens a pop-out window
  + In the left-hand window you can double click on “output.root” and then double click on “Events”
  + If you scroll down, you can double click on “n\_tracks”, which should then create a histogram with 100 entries, most of which should be 1. If there is a 1, then the muon was reconstructed, and if there is a 0, that means the muon was not reconstructed
  + To exit the root prompt, you have to enter “.q”
  + Then you could do something like “cp output.root output\_muons\_520.root” to preserve that first batch of events
* After you’ve looked at that and verified that most of the events have 1 track, could you then generate 100 muon gun events with a z vertex location of 700 cm?
  + Then open up the ntuple that gets generated and make the n\_tracks histogram
  + *In this case there should be no tracks*
  + This is the critical issue that we’re trying to address with your project. If the particles are created behind the first station of wire trackers, *then there are no tracks*
* Once you’ve done those things, let me know, and I will send further instructions about starting with the PID properly

**June 16, 2022 Updates**

**Step 1.0: Understanding the Reconstruction Code**

I think it’s a good idea to understand in a little detail how the simulation and reconstruction code works. Try to look through the flow I’ve outlined below, and check out the code snippets

1. First, physics events are generated. I.e. you can do a particle gun or JPsi or read in Aprime LHE files: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L222-L299>
2. Then we setup geant, which is how we simulate *particles interacting with the detector*: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L317-L353>
3. Digitization of simulated events - this is a simulation of how the *active* parts of the detector would respond to the particles passing through (“active” here means instrumented such that we can read out information): <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L356-L365>
4. **Registering reconstruction setup:** [**https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L379-L393**](https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L379-L393)
   1. **Note that the reconstruction class is “SQReco”**
5. Then some truth stuff: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L396-L405>
6. Then trigger emulation: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L407-L412>
7. Event filtering: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L414-L418>
8. Truth Vertexing: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L420-L424>
   1. I think this is where truth dimuon mass and z\_vtx is retrieved for example (if there are two truth-level muons in the event for instance)
9. RECO vetexing: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L426-L430>
10. Ntuple making: <https://github.com/wpmccormack/DarkQuest/blob/master/e1039-analysis/SimHits/macro/RecoE1039Sim.C#L432-L442>

**Step 1.1: The Reconstruction Step**

What we care about is the **reconstruction step. The following description is for generic tracks**

* The SQReco class is implemented here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx>
  + You can see the KalmanTracker assigned here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L132>
    - This is the algorithm that actually reconstructs tracks
  + The tracking is performed here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L360>
  + And tracks are retrieved here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L382>
  + The tracks that come out of the Kalman Tracker are in the “Tracklet” class. These only use drift chamber information, and have not actually had a “kalman” fit applied to them
  + These “Tracklet” tracks are converted into so called “SRecTrack”s here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L416>
    - (see <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L721> and <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L746>)
  + And then a RecTrack vector for the event is filled: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L445>

**Step 1.2: The Tracking Code**

The tracking algorithm that we use is pretty complicated, but I’ll try to give a quick overview. It is implemented here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx>

1. Straight-line tracklets are built in station 2 and station 3 separately: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx#L573-L595>
2. Station 2 and station 3 tracklets are combined to make station 2+3 tracklets: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx#L601-L830>
3. Tracklets are built in station 1 and then station 2+3 tracklets are connected to these tracks to create full tracks: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx#L893>

* A very important thing to note here is that there is “trackletsInSt” array which keeps track of tracklets at various points in the tracking code. It is referenced here for example: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.cxx#L830> and defined here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.h#L166>
  + You should also note that there is a function that can retrieve the tracklets at a given stage here: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/KalmanFastTracking_NEW_HODO_2.h#L139>
* Note that trackletsInSt[3] is the 2+3 tracklets, and trackletsInSt[4] is the full tracks

**Step 1.3: Existing Code for Station 3 Tracklets**

* Now, one “unfortunate” thing to note is that there were some recent changes to the tracking algorithm such that the station 2 and station 3 tracklets are created in a slightly different way than originally intended, so they are not stored in the trackletsInSt array by themselves. But as mentioned above, the station 2+3 tracklets are in trackletsInSt[3].
* *However*, before the tracking changes were made, I used to save the station 3 tracklets to our ntuples. In fact, if you open up the ntuples that you made for the particle guns earlier this week, I think you’ll find variables that have “st3tracklet” in the name. These should be empty right now, but I used to store station 3 tracklet information in them

1. If you search through this file for “st3”, you’ll find the things I needed to preserve that information: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.h>
2. Similarly, if you search through this file for “st3”, you’ll find all the things I had to do to full the st3 tracklet information: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx>
3. Then you can look through this file to see how I specified that st3 tracklet info should go in our ntuples: <https://github.com/wpmccormack/DarkQuest/blob/addTimer/e1039-analysis/SimHits/src/SimAna.h>
4. And lastly here you can see where I actually filled the information in the ntuples: <https://github.com/wpmccormack/DarkQuest/blob/addTimer/e1039-analysis/SimHits/src/SimAna.cc>

**Step 2: Saving at Station 2 + 3 Tracklets**

* For the displaced particle ID (PID), I am envisioning **potentially 2 types of displaced PID**: one using station 2+3 tracklets, and one using just station 3 tracklets alone
  + Using station 2+3 tracklets gives better position and angle measurements (for extrapolation) than using station 3 tracklets alone. However, this only works if the particles are created before station 2. This is a depth improvement over using full tracks of course, but it’s still not complete information
  + Using station 3 tracklets alone gives us the deepest range for PID, but the position and angle measurements from station 3 alone are not very good. There are also more fakes here
* To start out with, let’s do **displaced PID with station 2+3 tracklets.** This is the easier version of the two options

**CHECKPOINT 2:**

* The next big step in achieving displaced PID with station 2+3 tracklets is to save station 2+3 information in our ntuples
* What you should do here is basically duplicate the existing framework for station 3 tracklets, but do it for station 2+3 tracklets
  + You’ll need to look through the places I enumerated in step 1.3 above, and create similar lines for st23tracklets. E.g. create a line like this one: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.h#L156> and get the st23 tracklets with a line like this one: <https://github.com/wpmccormack/e1039-core/blob/patrick_new_tracking_displacedVertexing/packages/reco/ktracker/SQReco.cxx#L493>, where you getTrackletList(3)
* After you change the SQReco files, you need to go to <top directory where you set up the code>/e1039-core/ and compile with e.g.
  + ./build.sh -s packages/reco/ktracker
* You’ll also need to change the SimAna.cc and SimAna.h files. Once you do that, you need to navigate to <top directory where you set up the code>/e1039-core/DarkQuest/e1039-analysis/SimHits/work and do
  + make
  + make install
* The big deliverable here is to

1. Implement code to save the station 2+3 tracklets
2. Run a muon particle gun at z = 520 and look inside the ntuples. Is the station 2+3 tracklet information there? Does the position information match what you see for the regular tracks in this case?
3. Run a muon particle gun at z = 700 and look inside the ntuples. Now there shouldn’t be regular track information, but is there station 2+3 tracklet information? Hopefully there is!!