# Training Samples Preparation

mo.jia@stonybrook.edu

# Outline

- Framework
- Implementation on SeaWulf
- Test running results
- Next Steps

# Framework

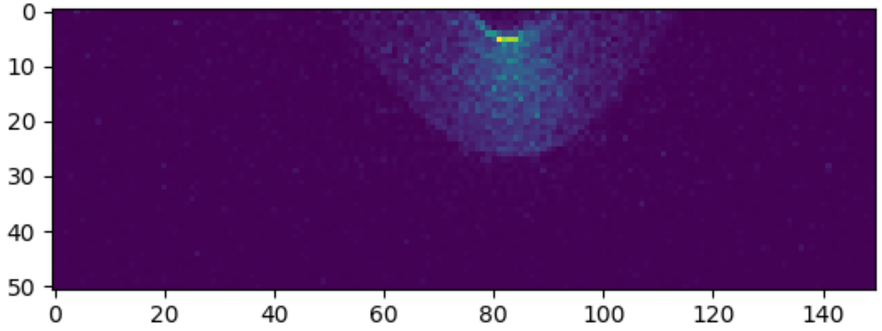
- Pyioopt
  - a python package to read WCSim root files and generate images of hits on the PMTs
  - Ultimately will be able to convert the data stored in root files to hdf5 format files, i.e. the training samples
  - Use pybind11 as the bindings to call c++ from python

# Implementation

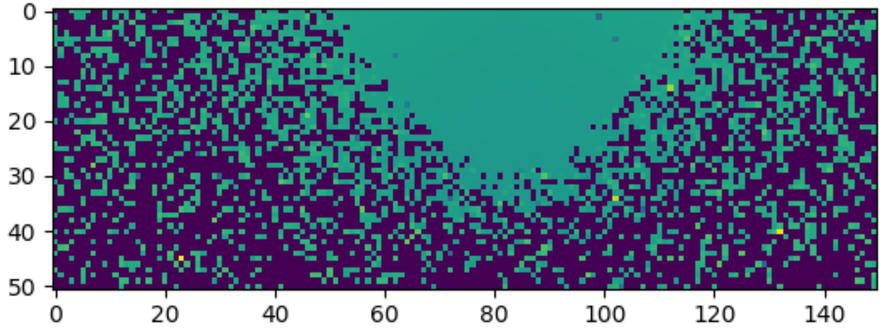
- Python3
- Root6
- WCSim
  - Set an environment variable ${WCSIMDIR}
- Append the directory of pyioopt to the list of ${PYTHONPATH}

# Test Running Results

Muon events in one root file

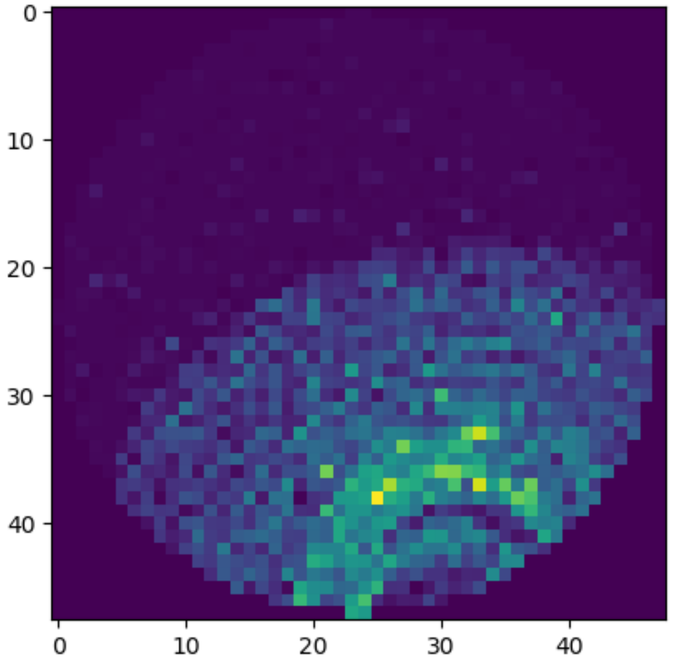



**Barrel q**

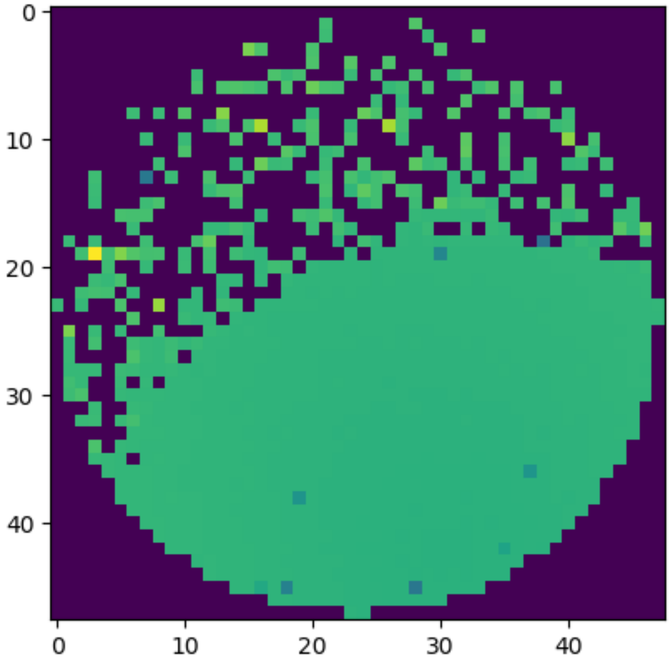



**Barrel t**

# Test Running Results

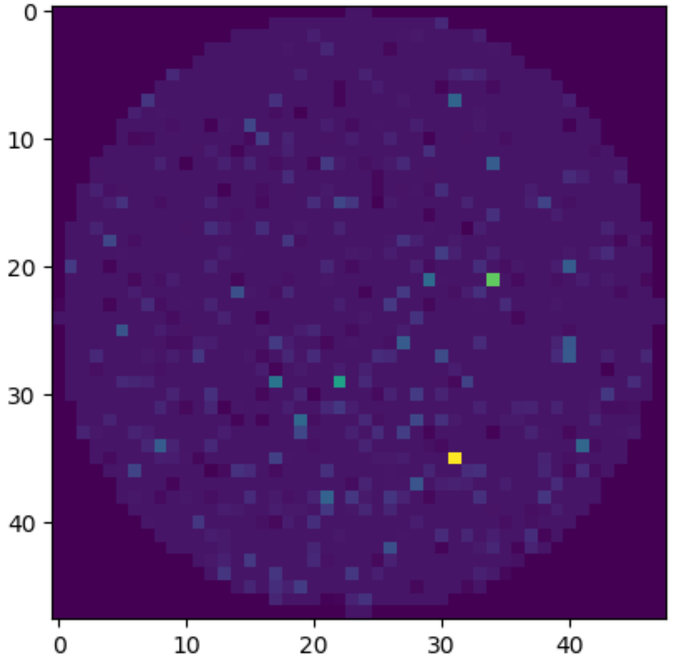



**Bottom q**

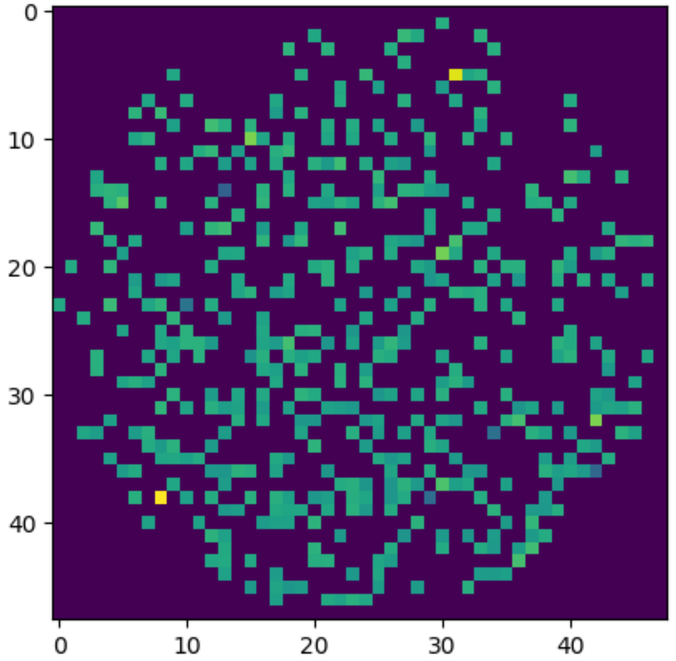



**Bottom t**

# Test Running Results




**Top q**




**Top t**

# Next Steps

- The ultimate goal is to convert the previous “images”, i.e. data on the grid, into hdf5 files with WatChMal format
- Study the DataTools package in WatChMal
- Learn more about hdf5 and the interface to it in python

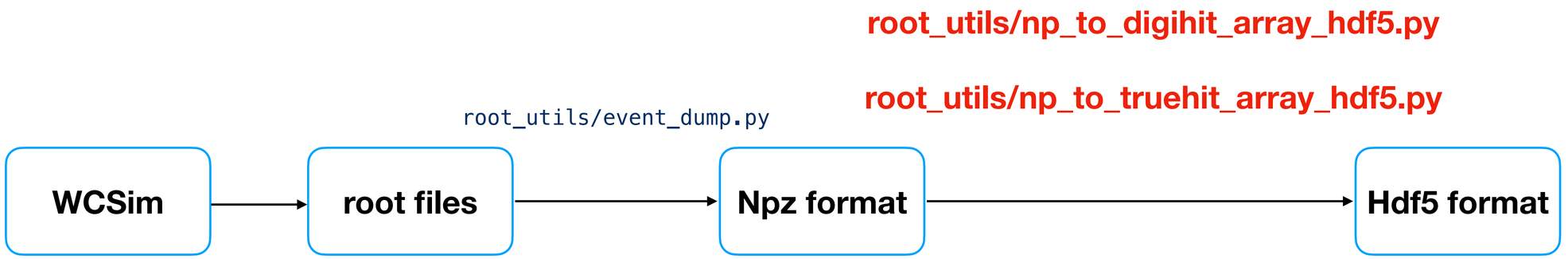
# DataTools Package in WatChMal

# DataTools Package

- Tools for production and manipulation of data for WatChMal
- Sub-directories:
  - data_quality
  - Visualization
  - cedar_scripts
  - root_utils

# Data Production for WatChMal

**Based on cedar_scripts/**




**run_WCSim_jobs.sh**

**make_digihit_h5.sh**

**make_truehit_h5.sh**

**NPZ** is a file **format** by numpy that provides storage of array data using gzip compression.

# root_utils/np_to_truehit_array_hdf5.py

- Create a h5py handle

```
20      config = get_args()
21      print("ouput file:", config.output_file)
22      f = h5py.File(config.output_file, 'w')
23
24      script_path = os.path.dirname(os.path.abspath(__file__))
25      git_status = subprocess.check_output(['git', '-C', script_path, 'status', '--porcelain', '--untracked-files=no']).decode()
26      if git_status:
27          raise Exception("Directory of this script ({}) is not a clean git directory:\n{}Need a clean git directory for storing script version in output file.".f
28      git_describe = subprocess.check_output(['git', '-C', script_path, 'describe', '--always', '--long', '--tags']).decode().strip()
29      print("git describe for path to this script ({}):".format(script_path), git_describe)
30      f.attrs['git-describe'] = git_describe
31      f.attrs['command'] = str(sys.argv)
32      f.attrs['timestamp'] = str(datetime.now())
33
```

# root_utils/np_to_truehit_array_hdf5.py

- Get total event numbers and hit numbers

```
34      total_rows = 0
35      total_hits = 0
36      print("counting events and hits in files")
37      for input_file in config.input_files:
38          print(input_file, flush=True)
39          if not os.path.isfile(input_file):
40              raise ValueError(input_file+" does not exist")
41          npz_file = np.load(input_file)
42          hit_pmts = npz_file['true_hit_pmt']
43          total_rows += hit_pmts.shape[0]
44          for h in hit_pmts:
45              total_hits += h.shape[0]
46
47      print(len(config.input_files), "files with", total_rows, "events with ", total_hits, "hits")
```

**total_rows = number of events**
**total_hits = number of hits**

# root_utils/np_to_truehit_array_hdf5.py

- Create datasets stored in the output file
  - Labels
  - root_files
  - event_ids
  - hit_time
  - hit_pmt
  - hit_parent
  - event_his_index
  - Energies
  - Positions
  - Angles
  - Veto
  - Veto2

```
49      dset_labels=f.create_dataset("labels",
50                                   shape=(total_rows,),
51                                   dtype=np.int32)
52      dset_PATHS=f.create_dataset("root_files",
53                                  shape=(total_rows,),
54                                  dtype=h5py.special_dtype(vlen=str))
55      dset_IDX=f.create_dataset("event_ids",
56                                shape=(total_rows,),
57                                dtype=np.int32)
58      dset_hit_time=f.create_dataset("hit_time",
59                                     shape=(total_hits, ),
60                                     dtype=np.float32)
61      dset_hit_pmt=f.create_dataset("hit_pmt",
62                                    shape=(total_hits, ),
63                                    dtype=np.int32)
64      dset_hit_parent=f.create_dataset("hit_parent",
65                                       shape=(total_hits, ),
66                                       dtype=np.int32)
67      dset_event_hit_index=f.create_dataset("event_hits_index",
68                                            shape=(total_rows,),
69                                            dtype=np.int64) # int32 is too small to fit large indices
70      dset_energies=f.create_dataset("energies",
71                                     shape=(total_rows, 1),
72                                     dtype=np.float32)
73      dset_positions=f.create_dataset("positions",
74                                      shape=(total_rows, 1, 3),
75                                      dtype=np.float32)
76      dset_angles=f.create_dataset("angles",
77                                   shape=(total_rows, 2),
78                                   dtype=np.float32)
79      dset_veto = f.create_dataset("veto",
80                                   shape=(total_rows,),
81                                   dtype=np.bool_)
82      dset_veto2 = f.create_dataset("veto2",
83                                    shape=(total_rows,),
84                                    dtype=np.bool_)
```

# root_utils/np_to_truehit_array_hdf5.py

- Read in data from npz file by file and set the values of datasets

```
    offset = 0
    offset_next = 0
    hit_offset = 0
    hit_offset_next = 0
    label_map = {22: 0, 11: 1, 13: 2}
    for input_file in config.input_files:
        print(input_file, flush=True)
        npz_file = np.load(input_file, allow_pickle=True)
        good_events = ~np.isnan(file_event_triggers[input_file])
        event_triggers = file_event_triggers[input_file][good_events]
        event_ids = npz_file['event_id'][good_events]
        root_files = npz_file['root_file'][good_events]
        pids = npz_file['pid'][good_events]
        positions = npz_file['position'][good_events]
        directions = npz_file['direction'][good_events]
        energies = npz_file['energy'][good_events]
        hit_times = npz_file['digi_hit_time'][good_events]
        hit_charges = npz_file['digi_hit_charge'][good_events]
        hit_pmts = npz_file['digi_hit_pmt'][good_events]
        hit_triggers = npz_file['digi_hit_trigger'][good_events]
        track_pid = npz_file['track_pid'][good_events]
        track_energy = npz_file['track_energy'][good_events]
        track_stop_position = npz_file['track_stop_position'][good_events]
        track_start_position = npz_file['track_start_position'][good_events]


        offset_next += event_ids.shape[0]

        dset_IDX[offset:offset_next] = event_ids
        dset_PATHS[offset:offset_next] = root_files
        dset_energies[offset:offset_next,:] = energies.reshape(-1,1)
        dset_positions[offset:offset_next,:,:] = positions.reshape(-1,1,3)

        labels = np.full(pids.shape[0], -1)
        for l, v in label_map.items():
            labels[pids==l] = v
        dset_labels[offset:offset_next] = labels

        polars = np.arccos(directions[:,1])
        azimuths = np.arctan2(directions[:,2], directions[:,0])
        dset_angles[offset:offset_next,:] = np.hstack((polars.reshape(-1,1),azimuths.reshape(-1,1)))
```

continued

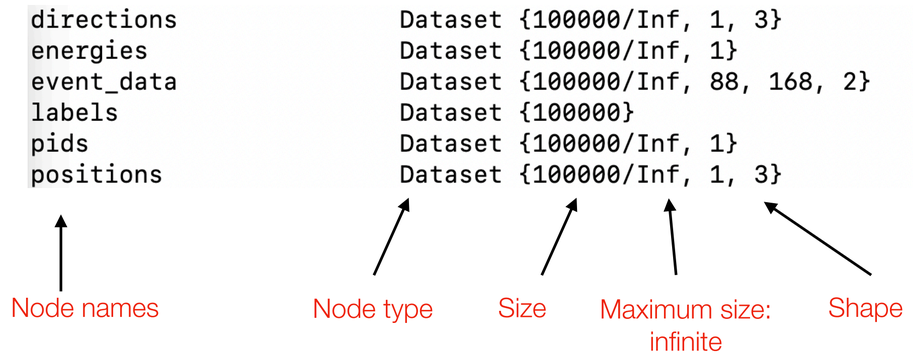
```
        for i, (pids, energies, starts, stops) in enumerate(zip(track_pid, track_energy,track_start_position, track_stop_position)):
            muons_above_threshold = (np.abs(pids) == 13) & (energies > 166)
            electrons_above_threshold = (np.abs(pids) == 11) & (energies > 2)
            gammas_above_threshold = (np.abs(pids) == 22) & (energies > 2)
            above_threshold = muons_above_threshold | electrons_above_threshold | gammas_above_threshold
            outside_tank = (np.linalg.norm(stops[:,(0,2)], axis=1) > config.radius) | (np.abs(stops[:, 1]) > config.half_height)
            dset_veto[offset+i] = np.any(above_threshold & outside_tank)
            end_energy_estimate = energies - np.linalg.norm(stops - starts, axis=1)*2
            muons_above_threshold = (np.abs(pids) == 13) & (end_energy_estimate > 166)
            electrons_above_threshold = (np.abs(pids) == 11) & (end_energy_estimate > 2)
            gammas_above_threshold = (np.abs(pids) == 22) & (end_energy_estimate > 2)
            above_threshold = muons_above_threshold | electrons_above_threshold | gammas_above_threshold
            dset_veto2[offset+i] = np.any(above_threshold & outside_tank)

        for i, (trigs, times, charges, pmts) in enumerate(zip(hit_triggers, hit_times, hit_charges, hit_pmts)):
            dset_event_hit_index[offset+i] = hit_offset
            hit_indices = np.where(trigs==event_triggers[i])[0]
            hit_offset_next += len(hit_indices)
            dset_hit_time[hit_offset:hit_offset_next] = times[hit_indices]
            dset_hit_charge[hit_offset:hit_offset_next] = charges[hit_indices]
            dset_hit_pmt[hit_offset:hit_offset_next] = pmts[hit_indices]
            hit_offset = hit_offset_next

        offset = offset_next
```

# Comparison with Current h5 files on Ivy

- Example: IWCDgrid_varyAll_mu-_20-2000MeV_100k.h5
- datasets stored in the file:
  - Directions shape (1,3 )
  - Energies shape (1, )
  - event_data shape (88,168,2)
  - Labels shape ?
  - Pid shape (1, )
  - Positions shape (1, 3)



# Next Steps

- Look into the codes in root_utils/event_dump.py
- Learn more about hdf5 and the interface h5py
- NumPy

# Recording WCSim-simulated Events Info to hdf5 Format

# h5py

- It is a Pythonic interface to the HDF5 binary data format
- Create a file object
    - f = h5py.File(‘myfile.h5’, ‘w’)
- Create datasets within an h5
    - dataset = f.create_dataset(‘node_name’, shape = (, ), dtype= )

# testWCSim.py in Pyioopt

- Pmt infos

```python
import wcsim_reader

import numpy as np
import matplotlib.pyplot as plt

rd = wcsim_reader.Reader()
print("Initialized reader")

rd.addFile("/gpfs/scratch/crfernandesv/TrainingSampleWCSim/mu-/1727/WCSim/out/*.root")

top = rd.mask[0]
barrel = rd.mask[1]
bottom = rd.mask[2]

print(len(rd))

print(rd.pmts())

for iev, event in enumerate(rd) :
    print("EVENT {0}".format(iev))
    print(event)
    for isub, sub in enumerate(event) :
        print ("SUB-EVENT {0}".format(isub))
        print(sub)
        print (sub["hits"])

        thisTop_q = np.copy(top).astype(float)
        thisTop_t = np.copy(top).astype(float)

        thisBarrel_q = np.copy(barrel).astype(float)
        thisBarrel_t = np.copy(barrel).astype(float)

        thisBottom_q = np.copy(bottom).astype(float)
        thisBottom_t = np.copy(bottom).astype(float)

        for hit in sub["hits"] :
            if rd.pmts()["location"][hit["pmtNumber"]-1] == 0 :
                thisTop_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
                thisTop_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
            elif rd.pmts()["location"][hit["pmtNumber"]-1] == 1 :
                thisBarrel_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
                thisBarrel_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
            elif rd.pmts()["location"][hit["pmtNumber"]-1] == 2 :
                thisBottom_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
                thisBottom_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
```

- Interface to wcsim root files in pyioopt
- Extract event information from root files after loading
- Each event is kept as an element in this iterable of reader object
- Event loop
  - Sub-event loop
- Hits loop

# Info kept by a reader object

```
15  struct pmt {
16    float x;
17    float y;
18    float z;
19    float dirx;
20    float diry;
21    float dirz;
22    int32_t location;
23    uint16_t row;
24    uint16_t column;
25  };
26
27  struct trueTrack {
28    int32_t PDG_code;
29    float m;
30    float p;
31    float E;
32    int32_t startVol;
33    int32_t stopVol;
34    float dirx;
35    float diry;
36    float dirz;
37    float stopx;
38    float stopy;
39    float stopz;
40    float startx;
41    float starty;
42    float startz;
43    int32_t parenttype;
44    float time;
45    int32_t id;
46  };
47
48  struct hit {
49    uint32_t pmtNumber;
50    float q;
51    float t;
52  };
53
54  struct vertex {
55    float vtx_x;
56    float vtx_y;
57    float vtx_z;
58  };
```

# Modification to testWCSim.py

```
1  import wcsim_reader
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  import h5py
7
8  rd = wcsim_reader.Reader()
9  print("Initialized reader")
10
11 rd.addFile("/gpfs/scratch/crfernandesv/TrainingSampleWCSim/mu-/1727/WCSim/out/*.root")
12
13 top = rd.mask[0]
14 barrel = rd.mask[1]
15 bottom = rd.mask[2]
16
17 print(len(rd))
18
19 print(rd.pmts())
20
21 output_file="/gpfs/scratch/mojia/HEPML/TrainingSampleWCSim/WCSim_mu_test.h5"
22
23 with h5py.File(output_file,'w') as f:
24     dset_directions=f.create_dataset("directions", shape=(len(rd), 1, 3), dtype=np.float32)
25     dset_energies=f.create_dataset("energies", shape=(len(rd), 1), dtype=np.float32)
26     dset_labels=f.create_dataset("labels", shape=(len(rd), ), dtype=np.int32)
27     dset_pids=f.create_dataset("pids", shape=(len(rd), 1), dtype=np.int32)
28     dset_positions=f.create_dataset("positions", shape=(len(rd), 1, 3), dtype=np.float32)
29     dset_event_data=f.create_dataset("event_data", shape=(len(rd), 51, 150, 2), dtype=np.float32)
30
31     for iev, event in enumerate(rd) :
32         print("EVENT {0}".format(iev))
33         print(event)
34
35         for isub, sub in enumerate(event) :
36             print ("SUB-EVENT {0}".format(isub))
37             print(sub)
38             print (sub["hits"])
39
40             vertex = sub["vertex"].copy()
41             vertex = vertex.view('<f4').reshape(1,3)
42             dset_positions[iev] = vertex
```

- Hard-corded shape, should be replaced by a flexible way
- (row, column)

# Modification to testWCSim.py

```
31      for iev, event in enumerate(rd) :
32          print("EVENT {0}".format(iev))
33          print(event)
34
35          for isub, sub in enumerate(event) :
36              print ("SUB-EVENT {0}".format(isub))
37              print(sub)
38              print (sub["hits"])
39
40              vertex = sub["vertex"].copy()
41              vertex = vertex.view('<f4').reshape(1,3)
42              dset_positions[iev] = vertex
43
44              thisTop_q = np.copy(top).astype(float)
45              thisTop_t = np.copy(top).astype(float)
46
47              thisBarrel_q = np.copy(barrel).astype(float)
48              thisBarrel_t = np.copy(barrel).astype(float)
49
50              thisBottom_q = np.copy(bottom).astype(float)
51              thisBottom_t = np.copy(bottom).astype(float)
52
53              for hit in sub["hits"] :
54                  if rd.pmts()["location"][hit["pmtNumber"]-1] == 0 :
55                      thisTop_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
56                      thisTop_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
57                  elif rd.pmts()["location"][hit["pmtNumber"]-1] == 1 :
58                      thisBarrel_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
59                      thisBarrel_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
60                  elif rd.pmts()["location"][hit["pmtNumber"]-1] == 2 :
61                      thisBottom_q[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['q']
62                      thisBottom_t[rd.pmts()["column"][hit["pmtNumber"]-1], rd.pmts()["row"][hit["pmtNumber"]-1]] = hit['t']
63
64              dset_event_data[iev,:,:,0] = thisBarrel_q
65              dset_event_data[iev,:,:,1] = thisBarrel_t
66
67              del thisTop_q, thisTop_t, thisBarrel_q, thisBarrel_t, thisBottom_q, thisBottom_t
68              break # break the sub loop, only keep record of the 0th subevent
69
70  #        break # just one event, for testing
71  print("Saved", iev+1, " events.")
```

- sub[“vertex”] is a structured array
- Conver it to normal ndarray type and reshape it to guarantee the compliance with the dataset

- Directly assign the charge and time 2D arrays to the dataset

# Test Results

**Tested with only one event**

```
495   DATASET "event_data" {
496      DATATYPE  H5T_IEEE_F32LE
497      DATASPACE  SIMPLE { ( 1, 51, 150, 2 ) / ( 1, 51, 150, 2 ) }
498      DATA {
499      (0,0,0,0): 1, 1,
500      (0,0,1,0): 1, 1,
501      (0,0,2,0): 1, 1,
502      (0,0,3,0): 1.0604, 1.0604,
503      (0,0,4,0): 13.2188, 13.2188,
504      (0,0,5,0): 0.427381, 0.427381,
505      (0,0,6,0): 5.76963, 5.76963,
506      (0,0,7,0): 1, 1,
507      (0,0,8,0): 1, 1,
508      (0,0,9,0): 0.770916, 0.770916,
509      (0,0,10,0): 1, 1,
510      (0,0,11,0): 0.442764, 0.442764,
511      (0,0,12,0): 1.20999, 1.20999,
512      (0,0,13,0): 1.10461, 1.10461,
513      (0,0,14,0): 1, 1,
514      (0,0,15,0): 1.12842, 1.12842,
515      (0,0,16,0): 1, 1,
516      (0,0,17,0): 1, 1,
517      (0,0,18,0): 0.204195, 0.204195,
518      (0,0,19,0): 1, 1,
519      (0,0,20,0): 1, 1,
520      (0,0,21,0): 0.969537, 0.969537,
```

```
8415     DATASET "positions" {
8416        DATATYPE  H5T_IEEE_F32LE
8417        DATASPACE  SIMPLE { ( 1, 1, 3 ) / ( 1, 1, 3 ) }
8418        DATA {
8419        (0,0,0): 636.794, 316.242, 622.826
8420        }
8421     }
8422  }
8423  }
```

# Problems

- Dataset for directions, pids, energies, labels
  - The “trueTracks” struct in the event object keeps the information of all tracks in that event, including the variables of interest
  - How to determine the variables of the event?
    - Which track should be considered as the one contains the information of the primary event ?
    - Or how to calculate the variables using all the tracks?
- Verification?