Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

Update on Architecture Review of Track3DKalmanHit Module

Saba Sehrish

LArSoft Coordination Meeting

02/16/16

LArSoft Architecture Review

- **Goal:** Interoperability of the code
  - The algorithm is separated from the module
  - There are interfaces to interact with the module
  - The code can be extended to enable interoperability
- **Candidate:** Track3DKalmanHit module
- **Analysis of the code:**
  - Complex algorithm and implementation
  - Implemented and tested for MicroBooNE only
  - 3 algorithm classes already exist
  - `produce` member function complex and hard to follow
    - > 600 lines of code + comments, which include all the logic of getting the input from an event, processing it, generating the output, and putting it back in an event
    - 3 different types of getting the input (hits) from an event

Task I: Setting up for the review work

- Make sure tests are in place to guarantee that physics meaning of code has not changed
  - An output module to dump tracks, space points and assns
  - A configuration file to run three input modes of the module
  - Compare the output of the original and refactored code
- Make sure the new code still passes the existing test
  - `mrb test`
  - `ctest -I 10, 10`
- Run through gdb and Valgrind if needed

Task II: Factorization of the `produce` function

- 17 new member functions were introduced to extract an algorithm from the module
- Most of the member functions will be moved to the new algorithm class
- The new `produce` method now follows the design as recommended by the *art* team.
- There are 5 distinct steps now in the `produce` method

```
1. void trkf::Track3DKalmanHit::produce(art::Event & evt) {
2. …some initializations
3.     getInputfromevent(evt, hits);
4.     generateKalmanTracks(hits, kalman_tracks);
5.     prepareOutput(kalman_tracks, tracks, spacepts, assns);
6.     populateEvent(evt, tracks, spacepts, assns);
7. }
```

Does not depend on an event

Create assns, depends on art event

Next steps for the Factorization work

- Factorization of the top level Track3DKalmanHit module will be completed by March 1st
  - A module that inherits from EDProducer
  - An algorithm class with well-defined member functions that will clearly describe the algorithm

- Tests will be provided

Task III: Track down errors

- Logical error in the Track3DKalmanHit module
  - There was a check in the end to only fill the tracks if numhits > 2
- Encountered EXC_BAD_ACCESS
  - Used Valgrind on Linux and libgmalloc with gdb on Mac
  - Running uboone stage 2 was always crashing in the debugger after enabling libgmalloc
    - There were two separate memory problems; larreco/TrackStitcher_module.cc and larana/CosmicRemoval/BeamFlashTrackMatchTaggerAlg.cxx
    - Two tickets issued (**11704, 11705**)
    - 11704 assigned to Saba Sehrish
    - 11705 should be taken care by the experiment

Task IV: Interoperability

- Initial analysis work for the **interoperability** has started, will be followed up after the Factorization is complete
- The current code has not been tested for experiments other than MicroBooNE
- Next steps:
  - More interviews will be conducted to gain an understanding for the use of geometry at the lowest level in the Kalman code
  - Identify pieces in the code to be extended for multiple TPCs
  - Implement
  - Set up tests for checking the correctness of the algorithm for different detectors
  - Several weeks worth of work

What more should be done

- *art* and LArSoft utilities
  - Identify patterns in the code that can generalized
- Performance profiling
  - Identify and address memory and CPU time performance
- Review of the existing algorithm classes of the Track3DKalmanHit module
- Review of the data structures that are involved
- Review of the linear algebra library used
- Many other modules can benefit from code review and refactoring