Fermilab

Art Mixing: Data and beyond!

Wesley Ketchum

Quick outline

- Why mix?
- Art mixing modules
  - Using the ubdataoverlay as an example
- References:
  - See github project site: https://github.com/wesketchum/ubdataoverlay

What?/why? Mixing

- Data mixing
  - MicroBooNE wants to mix MC neutrino interactions onto off-beam data events
    - Detector noise, detector cosmic rate, etc.
- MC Mixing
  - Improvements to background simulation generation
    - Like cosmics, “dirt” interactions, etc.

Implementation

- Art contains templated functions for “mixing filters”
- User specifies…
  - List of files containing events with products to add in
  - Number of events to add in
  - What order (sequential or random) to access events from those files
  - How to “mix” products together
  - Any additional information to put on the event
- See mu2e example from Rob Kutschke https://cdcvs.fnal.gov/redmine/projects/mu2eofflinesoftwaremu2eoffline/repository/revisions/master/entry/EventMixing/src/MixMCEvents_module.cc

Example
(ubdataoverlay/DataOverlayMixer/ OverlayRawDataMicroBooNE_module.cc)


```
38  namespace mix {
39    class OverlayRawDataDetailMicroBooNE;
40    typedef art::MixFilter<OverlayRawDataDetailMicroBooNE> OverlayRawDataMicroBooNE;
41  }
42
43  class mix::OverlayRawDataDetailMicroBooNE : public boost::noncopyable {
44  public:
45
46    OverlayRawDataDetailMicroBooNE(fhicl::ParameterSet const& p,
47                                   art::MixHelper &helper);
48
49    void startEvent(const art::Event&);  //called at the start of every event
50    void finalizeEvent(art::Event &);    //called at the end of every event
51
52    size_t nSecondaries() { return fEventsToMix; }
53
54    //void processEventIDs(art::EventIDSequence const& seq); //bookkepping for event IDs
55
56    // Mixing Functions
57
58    // For now, allow exactly one  input. Assume MC inputs have been merged
59    // previously and one detsim output created if needed. This could be changed
60    // but would require mixing functions for MC here.
61    bool MixRawDigits( std::vector< std::vector<raw::RawDigit> const* > const& inputs,
62                       std::vector<raw::RawDigit> & output,
63                       art::PtrRemapper const &);
64    /*
65    //TODO: OpDetWaveform
66    bool MixOpDetWaveform( std::vector< std::vector<raw::OpDetWaveform> const* > const& inputs,
67                           std::vector<raw::OpDetWaveform> & output,
68                           art::PtrRemapper const &);
69    */
70
71  private:
72
73    // Declare member data here.
```

Example
(ubdataoverlay/DataOverlayMixer/ OverlayRawDataMicroBooNE_module.cc)


```
38  namespace mix {
39    class OverlayRawDataDetailMicroBooNE;
40    typedef art::MixFilter<OverlayRawDataDetailMicroBooNE> OverlayRawDataMicroBooNE;
41  }
42
43  class mix::OverlayRawDataDetailMicroBooNE : public boost::noncopyable {
44  public:
45
46    OverlayRawDataDetailMicroBooNE(fhicl::ParameterSet const& p,
47                                   art::MixHelper &helper);
48
49    void startEvent(const art::Event&);  //called at the start of every event
50    void finalizeEvent(art::Event &);    //called at the end of every event
51
52    size_t nSecondaries() { return fEventsToMix; }
53
54    //void processEventIDs(art::EventIDSequence const& seq); //bookkepping for event IDs
55
56    // Mixing Functions
57
58    // For now, allow exactly one  input. Assume MC inputs have been merged
59    // previously and one detsim output created if needed. This could be changed
60    // but would require mixing functions for MC here.
61    bool MixRawDigits( std::vector< std::vector<raw::RawDigit> const* > const& inputs,
62                       std::vector<raw::RawDigit> & output,
63                       art::PtrRemapper const &);
64    /*
65    //TODO: OpDetWaveform
66    bool MixOpDetWaveform( std::vector< std::vector<raw::OpDetWaveform> const* > const& inputs,
67                           std::vector<raw::OpDetWaveform> & output,
68                           art::PtrRemapper const &);
69    */
70
71  private:
72
73    // Declare member data here.
```

Your class

The art class/module (which you don't touch!)

Example
(ubdataoverlay/DataOverlayMixer/ OverlayRawDataMicroBooNE_module.cc)


```
38  namespace mix {
39    class OverlayRawDataDetailMicroBooNE;
40    typedef art::MixFilter<OverlayRawDataDetailMicroBooNE> OverlayRawDataMicroBooNE;
41  }
42
43  class mix::OverlayRawDataDetailMicroBooNE : public boost::noncopyable {
44  public:
45
46    OverlayRawDataDetailMicroBooNE(fhicl::ParameterSet const& p,
47                                   art::MixHelper &helper);
48
49    void startEvent(const art::Event&);  //called at the start of every event
50    void finalizeEvent(art::Event &);    //called at the end of every event
51
52    size_t nSecondaries() { return fEventsToMix; }
53
54    //void processEventIDs(art::EventIDSequence const& seq); //bookkepping for event IDs
55
56    // Mixing Functions
57
58    // For now, allow exactly one  input. Assume MC inputs have been merged
59    // previously and one detsim output created if needed. This could be changed
60    // but would require mixing functions for MC here.
61    bool MixRawDigits( std::vector< std::vector<raw::RawDigit> const* > const& inputs,
62                       std::vector<raw::RawDigit> & output,
63                       art::PtrRemapper const &);
64    /*
65    //TODO: OpDetWaveform
66    bool MixOpDetWaveform( std::vector< std::vector<raw::OpDetWaveform> const* > const& inputs,
67                           std::vector<raw::OpDetWaveform> & output,
68                           art::PtrRemapper const &);
69    */
70
71  private:
72
73    // Declare member data here.
```

Called at the start of every event

Called at the end of every event

Determines how many events to mix in from source files (could follow Poisson!)

Your mixing functions (following that exact signature!)

The usual private class stuff you need

Example: fcl snippet

```
53   filters : {
54             mixer: { module_type : OverlayRawDataMicroBooNE
55                      fileNames : [ " prod_pions_detsim.root" ]
56                      readMode : sequential
57                      wrapFiles : true
58                      coverageFraction : 1
59                      detail : {
60                        RawDigitInputModuleLabel : daq
61                        EventsToMix: 1
62                        DefaultMCScale: 1.0
63                      }
64                    }
```

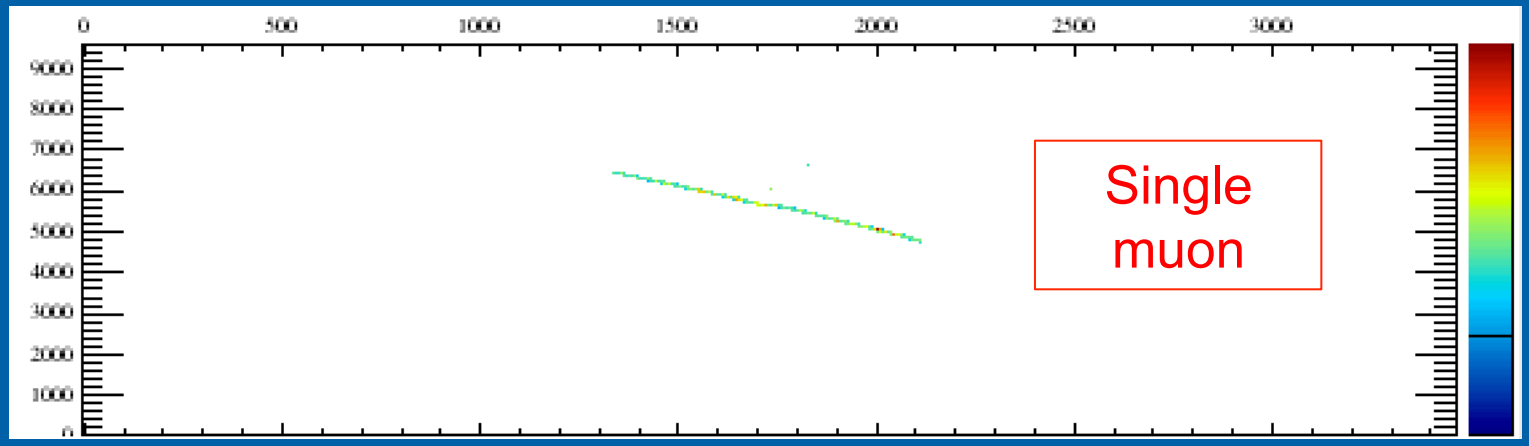
Mixing module specifics

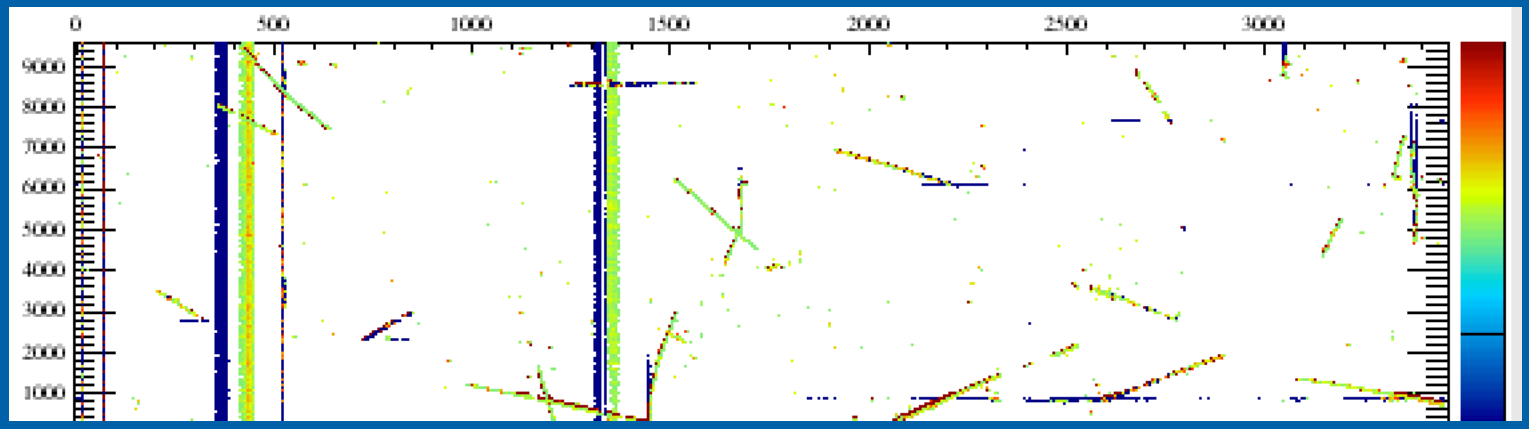
- Order in which things are called in event
  - startEvent()
    - Initialization…
  - nSecondaries()
    - How many events to mix
  - processEventIDs()
    - Details on the events you pull from mixing files
    - NOTE: you must have unique event IDs (run, subrun, event numbers)
  - Methods registered by declareMixOp, in the order you registered them
  - finalizeEvent()

UB TPC data overlay details

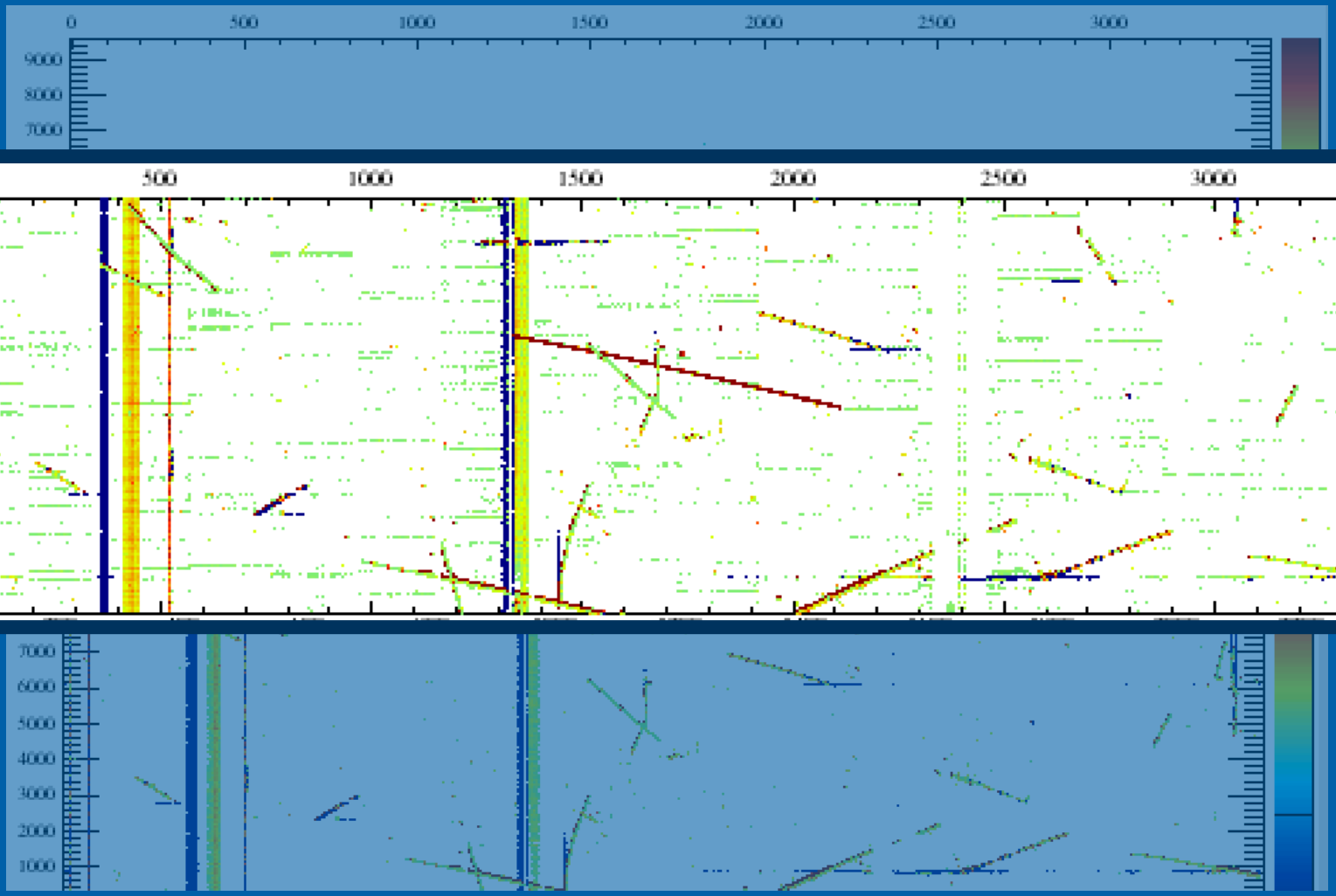
- Module startEvent
  - Can use data or MC file as the input file, and the other must be in the mixing file list
  - “Generates” scaling factors to apply to MC raw digits
    - Right now, defaults to 1 for all channels
    - Except … gets channel status, and assigns 0 for dead channels
      - Calibration → non-zero/non-unity factors in the future?
- Mixing work done by RDMixer class
  - Module enforces to one event to mix
  - Ignores MC contributions from channels not in data collection
  - Aligns start of MC waveform to start of data waveform
    - Assigns pedestal and sigma of data channel to new raw digit

Results…

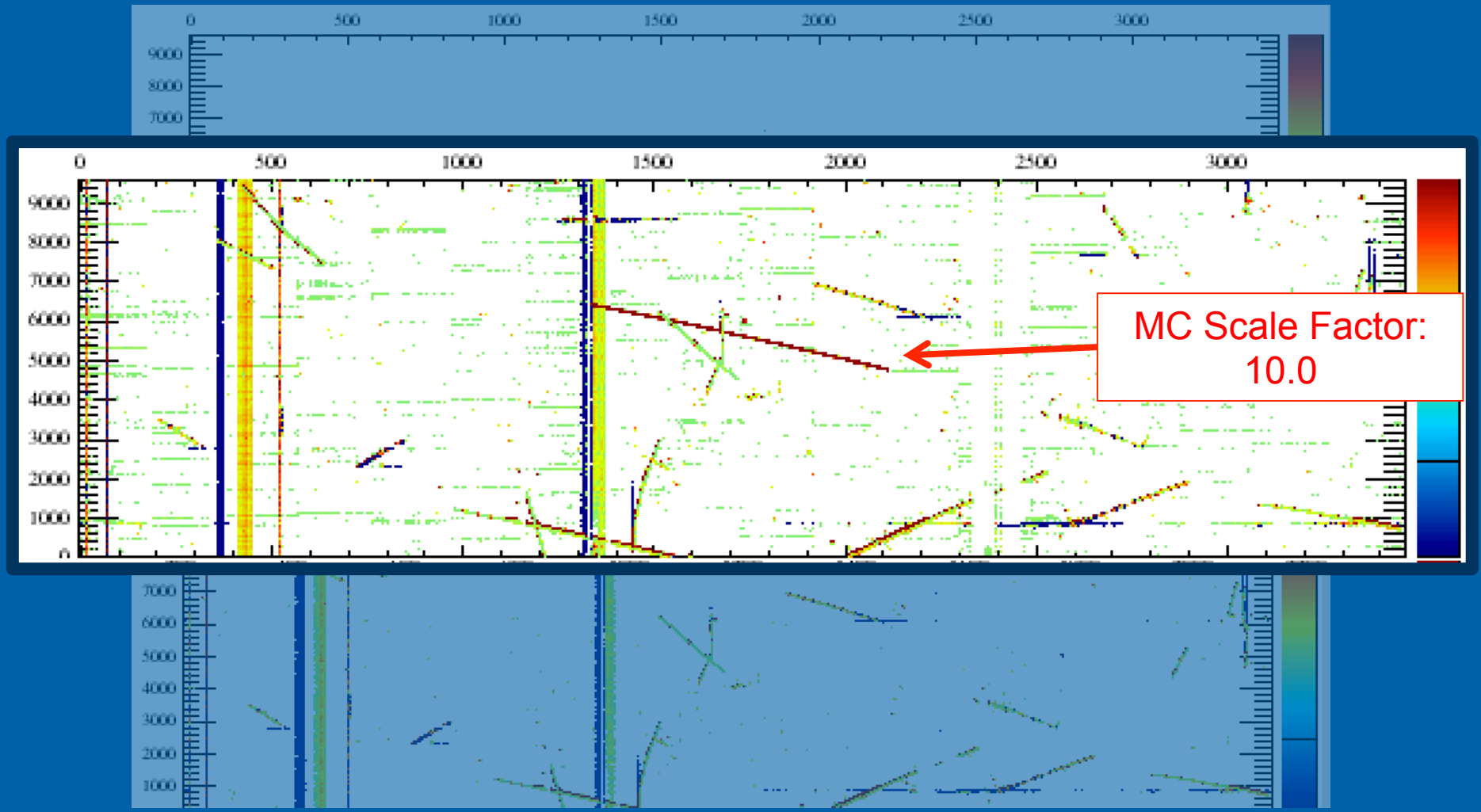

0
500
1000
1500
2000
2500
3000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Single muon


0
500
1000
1500
2000
2500
3000
9000
8000
7000
6000
5000
4000
3000
2000
1000

Results!


0
500
1000
1500
2000
2500
3000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

Results!



Current hangups/issues/notes

- You need to explicitly copy collections from mixing files if you want them included
  - I wrote simple templated function that works
- Associations need to be remade
  - There is a PtrRemapper utility provided, which does much of the work, but there is still more to do
  - In discussion with artists about requesting this feature
  - Decided to mix data event onto MC for this reason
- Currently some small trouble with putting info onto the event
  - Not sure why…probably something I'm doing wrong

Where next (1)

- Updates from past few days
  - MC information all being included in mixed event
  - Channel status information
  - Incorporating optical data
  - See remaining issues: https://github.com/wesketchum/ubdataoverlay/issues
- Different modes for mixing?
  - Could mix in MC and data event together, with some empty or specialized "mix info" source
- Production routine will probably be next hurdle for MicroBooNE in relation to data overlays
  - I'm sure we'll share anything we uncover on best practices

Where next (2)

- This all lives in its own project
  - Not even in uboonecode(!), though the plan will be to merge it there in next few days
  - QUESTION: who's interested in a generic data overlay utility, and what are the ideas with respect to that?
    - Who wants to work on it? ☺
- Mixing modules could be used for cosmic ray generation/other backgrounds
  - Currently track every particle in large bounding box → heavy memory use
  - Could allow us to launch large cosmic-ray generation project across OSG
    - Mixing would involve varying particles in time?
  - Could also allow us more freedom in what level we track/bounding box to track per job
  - Could allow us to update cosmic rate later based on data
    - Or do studies with cosmic rate as systematic
  - QUESTION: who's interested? We may attack this in next month…