recob::Wire Modifications

Bruce Baller
March 26, 2014

Outline

- Motivation for changing recob::Wire
- Regions Of Interest (ROIs) concept
- Implementation for MicroBooNE
- Pros and Cons

- Slides for later discussion
  - Ruminations on other RecoBase objects

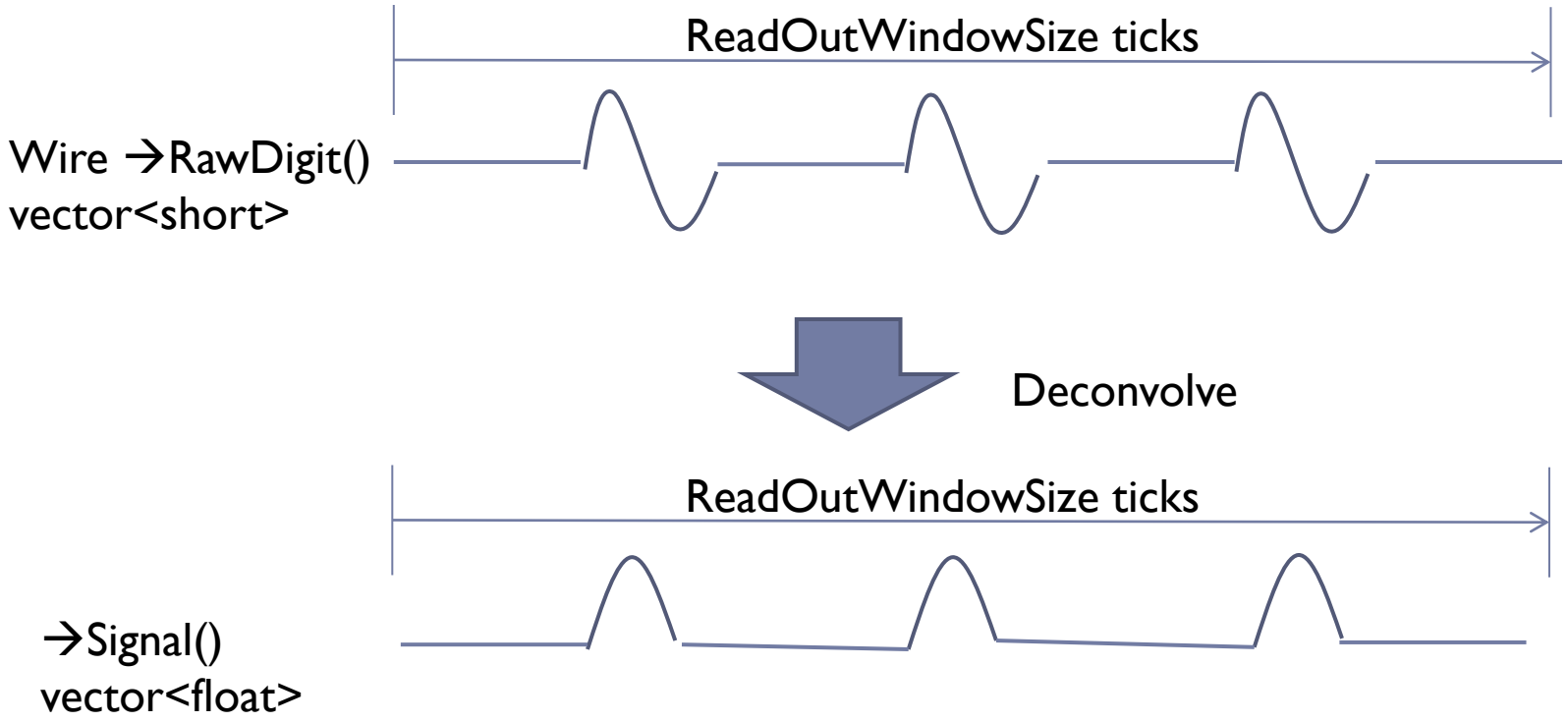
Motivation

- Wire signal and raw signal data products contain the full readout window but the non-zero signal occupancy (aka Region of Interest) is low
- Current scheme is to deconvolve RawDigits in all time bins on each wire
  - FFT requires data to be put in an array of size 2^N complex numbers – complex doubles since we are using ROOT
    - For MicroBooNE
      - Single 3200 tick readout frame → 2 x 4096 numbers
      - Single 9600 tick readout frame → 2 x 16384 numbers
- Lots of unnecessary computation and storage

Current & Proposed Schemes

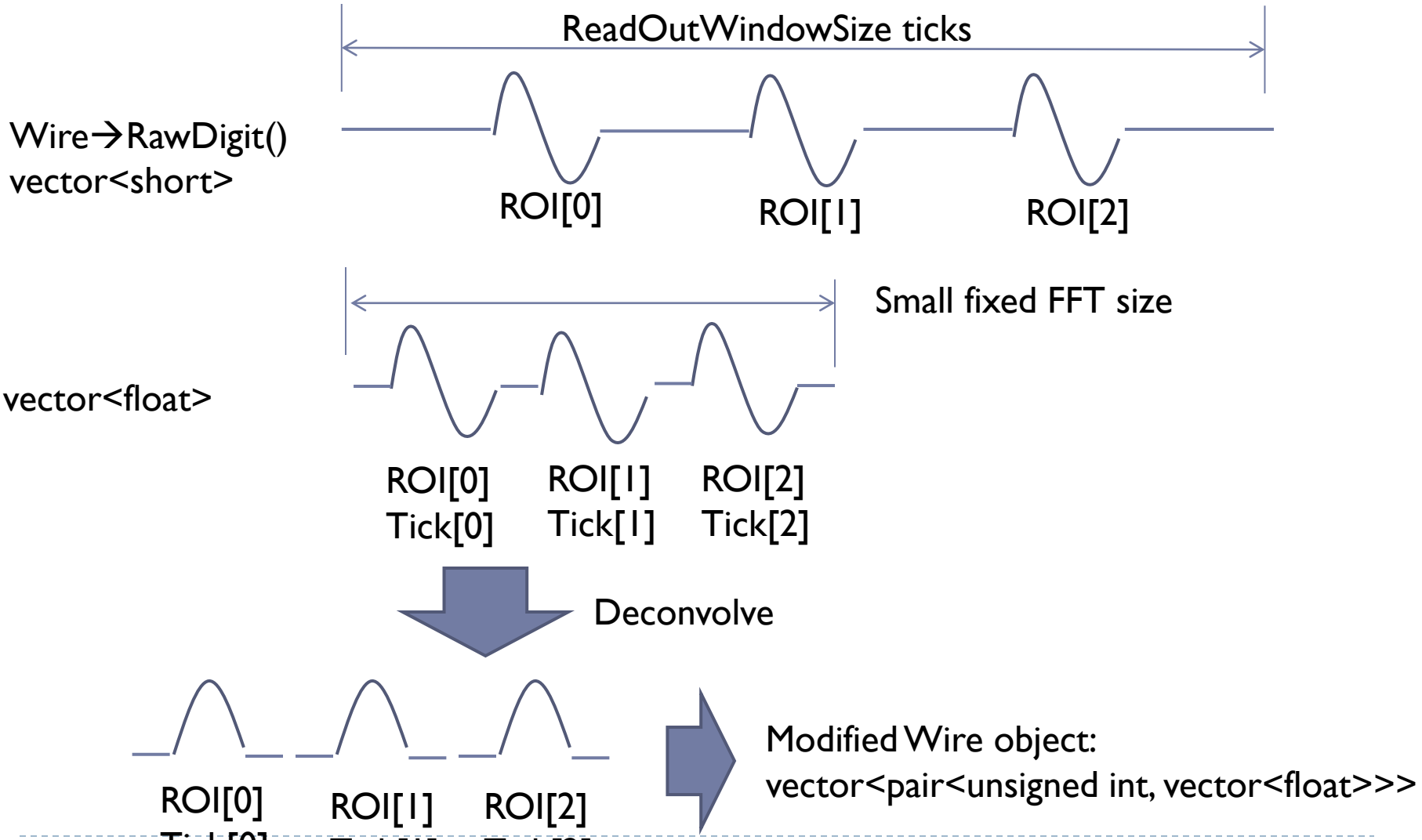
- Current scheme
  - CalWire
    - Initialize FFT service with FFTSize = ReadOutWindowSize
    - Deconvolve RawDigits and create recob::Wire fSignal
  - HitFinder
    - Find Signal regions above threshold
    - Fit to N Gaussians and create recob::Hits
- Proposed scheme
  - CalWire
    - Initialize FFT service with fcl file selectable FFTSize
    - Find fabs(RawDigit) regions above threshold (= ROIs)
    - Deconvolve ROIs and create Wire→SignalROI()
  - HitFinder – 2 options
    - Modify: Fit to N Gaussians using SignalROIs
    - Don't modify: Use Wire→Signal() method to get a zero padded signal vector of length ReadOutWindowSize

Wire Signal – Current Scheme



Note:
ALL wire planes are deconvolved
ALL wire plane signals have ReadOutWindowSize ticks

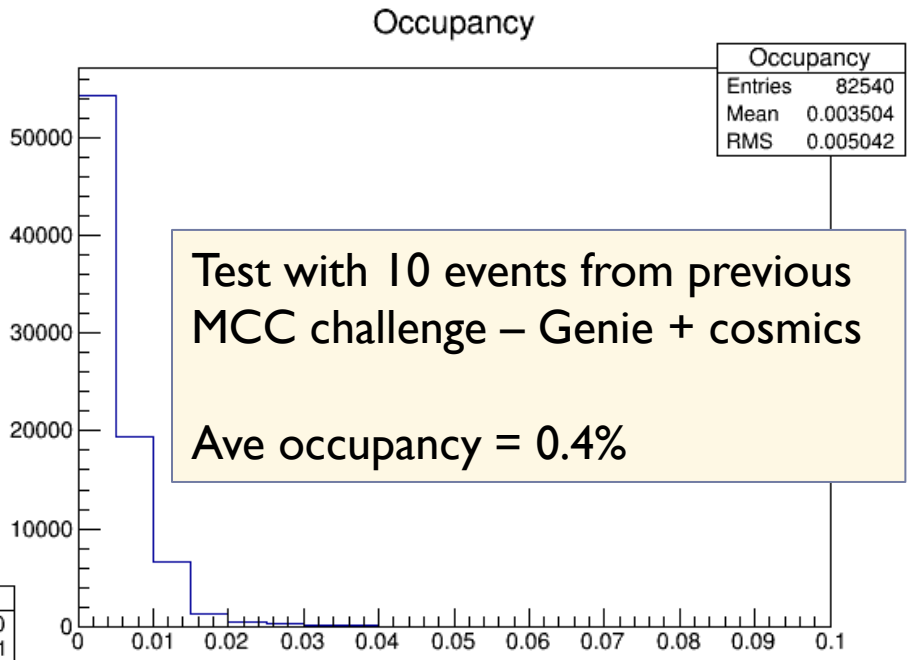
Wire Signal ROI Scheme



What fraction of the Signal contains hit information? - MicroBooNE

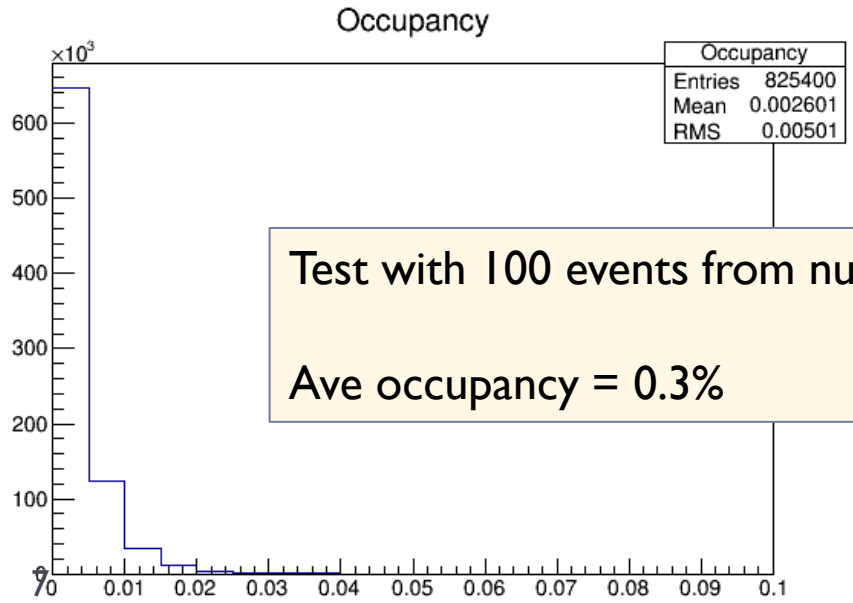
totLen = Σ bin length of all ROIs on a wire

Occupancy per wire = totLen / dataSize



Test with 10 events from previous MCC challenge – Genie + cosmics

Ave occupancy = 0.4%



Test with 100 events from nue_cosmic_3window

Ave occupancy = 0.3%

Wire.h (read only) - /grid/fermiapp/products/larsoft/lardata/nightly/source/RecoBase/ <@uboonegpvm06.fnal.gov>

File Edit Search Preferences Shell Macro Windows Help

```
private:

      std::vector<float>       fSignal;     ///< the calibrated signal waveform
      art::Ptr<raw::RawDigit> fRawDigit;    ///< vector to index of raw digit for this wire
      geo::View_t              fView;       ///< view corresponding to the plane of this wire
      geo::SigType_t           fSignalType; ///< signal type of the plane for this wire

#ifndef __GCCXML__

  public:
      Wire(std::vector<float> siglist,
           art::Ptr<raw::RawDigit> &rawdigit);

      // Get Methods
      const std::vector<float>&  Signal()     const;
      size_t                     NSignal()    const;
      art::Ptr<raw::RawDigit>    RawDigit()   const;
      geo::View_t                View()       const;
      geo::SigType_t             SignalType() const;
      uint32_t                   Channel()    const;

#endif

    };
}

#ifndef __GCCXML__

inline const std::vector<float>&  recob::Wire::Signal()     const { return fSignal;              }
inline size_t                     recob::Wire::NSignal()    const { return fSignal.size();       }
inline art::Ptr<raw::RawDigit>    recob::Wire::RawDigit()   const { return fRawDigit;            }
inline geo::View_t                recob::Wire::View()       const { return fView;                }
inline geo::SigType_t             recob::Wire::SignalType() const { return fSignalType;          }
inline uint32_t                   recob::Wire::Channel()    const { return fRawDigit->Channel();}

#endif

#endif // WIRE_H

//////////////////////////////////////////////////////////////////////////
```

Wire.h Current

Wire.h - /uboone/app/users/baller/roi/srcs/lardata/RecoBase/ <@uboonegpvm05.fnal.gov>

File Edit Search Preferences Shell Macro Windows Help

```
private:
      std::vector< std::pair< unsigned int, std::vector<float> > > fSignalROI;
      art::Ptr<raw::RawDigit> fRawDigit;    ///< vector to index of raw digit for this wire
      geo::View_t             fView;        ///< view corresponding to the plane of this wire
      geo::SigType_t          fSignalType; ///< signal type of the plane for this wire
      unsigned int            fMaxSamples; ///< max number of ADC samples possible on the wire

#ifndef __GCCXML__

  public:

      // ROI constructor
      Wire(std::vector< std::pair< unsigned int, std::vector<float> > > sigROIlist,
           art::Ptr<raw::RawDigit> &rawdigit);

      // Get Methods
      // zero-padded full length vector filled with ROIs
      std::vector<float>  Signal() const;

      const std::vector< std::pair< unsigned int, std::vector<float> > >& SignalROI()  const;
      size_t                    NSignal()    const;
      art::Ptr<raw::RawDigit>   RawDigit()   const;
      geo::View_t               View()       const;
      geo::SigType_t            SignalType() const;
      uint32_t                  Channel()    const;

#endif

    };
}

#ifndef __GCCXML__

inline const std::vector< std::pair< unsigned int, std::vector<float> > >& recob::Wire::SignalROI
  const { return fSignalROI;}
inline size_t                   recob::Wire::NSignal()    const { return fMaxSamples;          }
inline art::Ptr<raw::RawDigit>  recob::Wire::RawDigit()   const { return fRawDigit;            }
inline geo::View_t              recob::Wire::View()       const { return fView;                }
inline geo::SigType_t           recob::Wire::SignalType() const { return fSignalType;          }
inline uint32_t                 recob::Wire::Channel()    const { return fRawDigit->Channel();}
```

Wire.h Proposed

Wire.cxx - /uboone/app/users/baller/roi/srcs/lardata/RecoBase/ <@uboonegpvm05.fnal.gov>

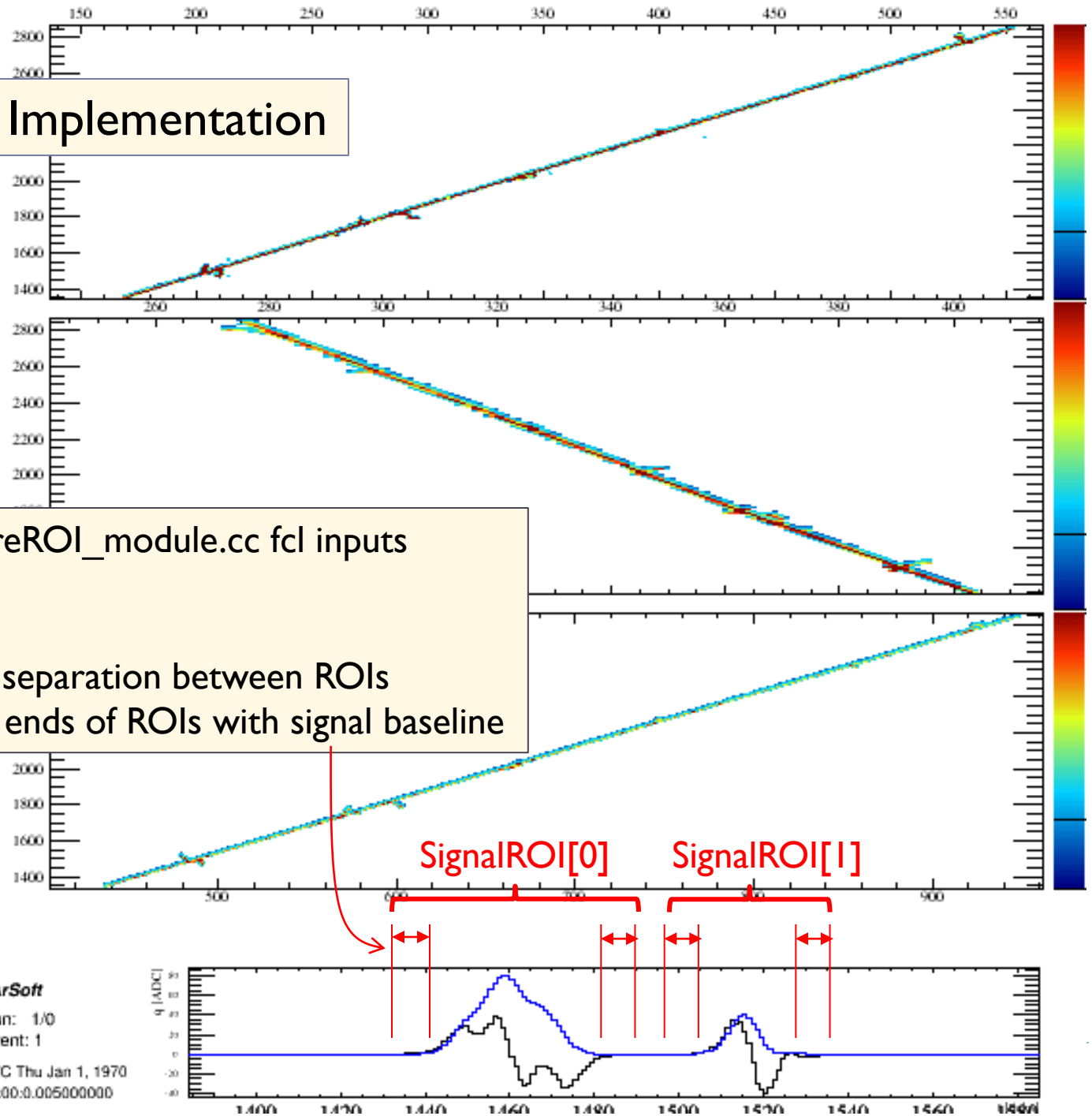
File Edit Search Preferences Shell Macro Windows Help

Wire.cxx | Wire.h

```
namespace recob{

  //-----------------------------------------------------------------------
  Wire::Wire()
    : fSignalROI(0)
  {

  }

  //-----------------------------------------------------------------------
  Wire::Wire(
    std::vector< std::pair< unsigned int, std::vector<float> > > sigROIlist,
    art::Ptr<raw::RawDigit> &rawdigit)
    : fSignalROI(sigROIlist)
    , fRawDigit(rawdigit)
  {

    art::ServiceHandle<geo::Geometry> geo;

    fView       = geo->View(rawdigit->Channel());
    fSignalType = geo->SignalType(rawdigit->Channel());
    fMaxSamples = rawdigit->NADC();

  }

std::vector<float> Wire::Signal() const
{
  // Return ROI signals in a zero padded vector of size that contains
  // all ROIs

  std::vector<float> sigTemp(fMaxSamples, 0.);
  if(fSignalROI.size() == 0) return sigTemp;

  for(unsigned int ir = 0; ir < fSignalROI.size(); ++ir) {
    unsigned int tStart = fSignalROI[ir].first;
    for(unsigned int ii = 0; ii < fSignalROI[ir].second.size(); ++ii)
      sigTemp[tStart + ii] = fSignalROI[ir].second[ii];
  } // ir

  return sigTemp;
```

Wire.cxx Proposed

MicroBooNE Implementation



uboonecode/CalWireROI_module.cc fcl inputs
fThreshold (= 3)
fMinWid (= 4)
fMinSep (= 10) min separation between ROIs
fROIPad (= 10) pad ends of ROIs with signal baseline

CalWireT962_module.cc - /uboone/app/users/baller/roi/srcs/larevt/CalData/ <@uboonegpvm05.fnal.gov>

File Edit Search Preferences Shell Macro Windows Help

```
          expFit.SetRange(dataSize,transformSize);
          for(bin = 0; bin < dataSize; ++bin)
            holder[dataSize+bin]= expFit.Eval(bin+dataSize);
        }
        // This is actually deconvolution, by way of convolution with the inverted
        // kernel.  This code assumes the response function has already been
        // been transformed and inverted.  This way a complex multiplication, rather
        // than a complex division is performed saving 2 multiplications and
        // 2 divsions

        fFFT->Convolute(holder,kernel[kernMap[channel]]);
      } // end if channel is a good channel

      holder.resize(dataSize,1e-5);
      //This restores the DC component to signal removed by the deconvolution.
      if(fPostsample) {
        double average=0.0;
        for(bin=0; bin < (unsigned int)fPostsample; ++bin)
          average+=holder[holder.size()-1-bin]/(double)fPostsample;
        for(bin = 0; bin < holder.size(); ++bin) holder[bin]-=average;
      }

      // Make a single ROI that spans the entire data size
      std::vector<std::pair<unsigned int, std::vector<float>>> hvec;
      hvec.push_back(std::make_pair(0, holder));
      wirecol->push_back(recob::Wire(hvec,digitVec));
    }

    if(wirecol->size() == 0)
      mf::LogWarning("CalWireT962") << "No wires made for this event.";

    evt.put(std::move(wirecol));

    delete chanFilt;
    return;
  }

} // end namespace caldata

namespace caldata{
```

ArgoNeuT CalWire modification → one big ROI per wire

Change CCHitFinder to use ROIs

CCHitFinderAlg.cxx - /uboone/app/users/baller/roi/srcs/larreco/RecoAlg/ <@uboonegpvm05.fnal.gov>

File Edit Search Preferences Shell Macro Windows

```
      // get the Regions Of Interest on this wire
      std::vector<std::pair< unsigned int, std::vector<float> > >
        ROIs(theWire->SignalROI());

      // loop over the ROIs
      for(unsigned short ir = 0; ir < ROIs.size(); ++ir) {
        bumps.clear();
        unsigned short tStart = ROIs[ir].first;
        unsigned short npt = ROIs[ir].second.size();
        // transfer the ROI into the signal array
        for(unsigned short ii = 0; ii < npt; ++ii) {
          signl[ii] = ROIs[ir].second[ii];
          // look for bumps
          if(ii > 3) {
            if(signl[ii - 2] > minSig) {
              if(signl[ii - 2] > signl[ii - 3] &&
                 signl[ii - 3] > signl[ii - 4] &&
                 signl[ii - 2] > signl[ii - 1] &&
                 signl[ii - 1] > signl[ii]) bumps.push_back(ii - 2);
            } // signl[ii - 2] > minSig
          } // ii > 3
        } // ii
        // make a crude hit if there are too many bumps
        if(bumps.size() > fMaxBumps) {
          MakeCrudeHit(npt, ticks, signl);
          StoreHits(tStart, npt, theWire);
          continue;
        } // bumps.size() > fMaxBumps
        // start looking for hits with the found bumps
        unsigned short nHitsFit = bumps.size();
        unsigned short nfit = 0;
        chidof = 0.;
        bool HitStored = false;
        unsigned short nMaxFit = bumps.size() + fMaxXtraHits;
//      bool first = true;
        while(nHitsFit <= nMaxFit) {
          FitNG(nHitsFit, npt, ticks, signl);
```

Get the ROIs on the wire

Find N bumps within the ROI

Fit to N Gaussians

Summary

- Pros
  - Significant reduction in memory and file size using ROIs
  - New Signal get method returns a zero-padded vector ala the old Wire object
    - Minimal changes to event display & hit finders
  - The "bump hunting" code in the hit finders can be eliminated if the SignalROI get method is used
    - HitFinder → HitFitter
- Cons
  - A means of reading/converting existing MC files is needed if this is deemed to be a requirement – is it?
    - Alternatively, one could read existing MC files with v1_00_05

Ruminations on other RecoBase objects

- Use "graded approach" when considering changes to add or remove features
  - Roughly speaking …
    - Thousands of hits per event → be hard-nosed
    - Hundreds of clusters per event
    - Tens of tracks per event → be loose

Ruminations on recob::Hit

- Used
  - PeakTime → x
  - TotCharge → dQ/dx
  - σ = EndTime - PeakTime
  - Multiplicity, GoodnessOfFit
- Confusing
  - maxCharge = amplitude
  - totCharge = $\sqrt{2\pi}\,\sigma$ Amp
- Not filled/used/needed
  - Sigma…

```
Hit::Hit(art::Ptr<raw::RawDigit> rawdigit,
         geo::View_t view,
         geo::SigType_t signaltype,
         geo::WireID wid,
         double startTime, double sigmaStartTime,
         double endTime,   double sigmaEndTime,
         double peakTime,  double sigmaPeakTime,
         double totcharge, double sigmaTotCharge,
         double maxcharge, double sigmaMaxCharge,
         int    multiplicity,
         double goodnessOfFit
         )
```

- Float has sufficient precision
  - Hit position resolution > 200 μm (x_{max} ~ 250.02 cm)
  - Wire-to-wire ionization fluctuations are large ~30%

Ruminations on recob::Cluster

Track-like clusters…

```
double                          fTotalCharge;     ///< total charge in cluster
double                          fdTdW;            ///< slope of cluster in tdc vs wire
double                          fdQdW;            ///< slope of cluster in charge vs wire
double                          fSigmadTdW;       ///< slope of cluster in tdc vs wire
double                          fSigmadQdW;       ///< slope of cluster in charge vs wire
std::vector<double>             fStartPos;        ///< start of cluster in (wire, tdc) plane
std::vector<double>             fEndPos;          ///< start of cluster in (wire, tdc) plane
std::vector<double>             fSigmaStartPos;   ///< start of cluster in (wire, tdc) plane
std::vector<double>             fSigmaEndPos;     ///< start of cluster in (wire, tdc) plane
int                             fID;              ///< cluster's ID
geo::View_t                     fView;            ///< view for this cluster
```

- Not useful
  - fTotalCharge
  - fdQdW (varies)
- Cluster slope
  - Start dTdW ≠ End dTdW
- Cluster charge at Start/End would be useful for 3D track matching
  - ClusterCrawler defines Begin == end of the cluster with the lower charge
- Float has adequate precision – ala Hit