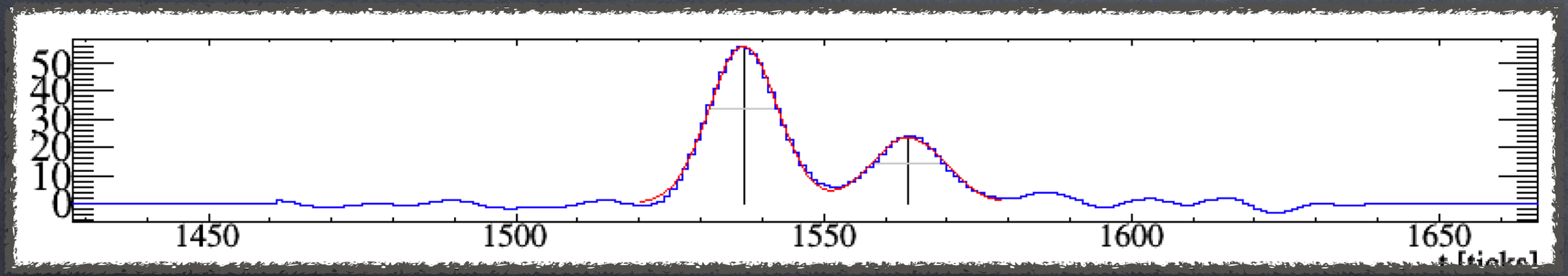


GausHitFinder

Hit Finding and Fitting

Tracy Usher

Workshop on Calibration and Reconstruction
for LArTPC Detectors

December 11, 2018

Motivation

- LArSoft reconstruction is currently hit based, finding hits represents the final step in the series of signal processing tasks
- The GausHitFinder is the primary hit finding module
  - It is probably the only hit finder operating on deconvolved waveforms
- The task of GausHitFinder remains the same, the structure has evolved considerably from its initial version of many years ago
  - Its more modular
  - There have been changes reflecting requests from users
  - There are also updates to keep pace with underlying code
- There is currently no single overview of how the module works, what options exist, how to control the parameters, etc.
- The principle goal of this presentation is to try to rectify this situation.

GausHitFinder

Original Version

- Assumes that the input waveform contains one or more gaussian shaped pulses signifying hits
  - Expected result from the deconvolution procedure
- GausHitFinder searches input waveforms to find candidate hits and then fit one or more gaussians to return reconstructed hits
  - Note: Hits can overlap (e.g. delta ray production)
- Code all contained in one module, all work done in a loop over input deconvolved waveforms
- Original module authored by Jonathan Asaadi in the early days of LArSoft

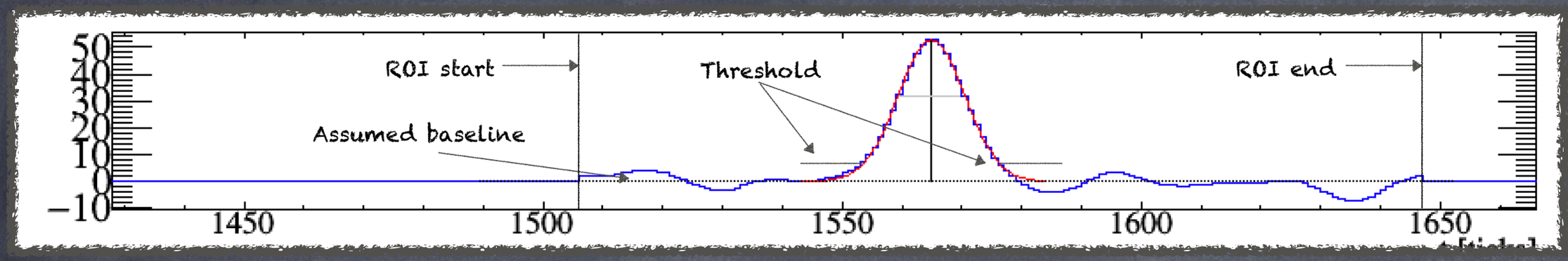
GausHitFinder

Primary Architecture Change

- Advent of art tools allowed GausHitFinder to be refactored:
  - Separate "candidate peak finding" from "peak fitting"
  - These Tasks re-implemented with art tools via interface classes
    - See the LArReco/HitFinder/HitFinderTools folder for their definitions
- The tool interfaces facilitate interchangeability of the underlying algorithms for hit finding and fitting
  - e.g. Currently have three methods for hit finding available in LArReco
- Note: in addition to the currently available tools this interface also allows the ability for experiments to implement their own tools to better handle issues specific to their detector
- Also note: the above changes also require the fhicl files change

Quick Overview of Peak Finding Algorithms

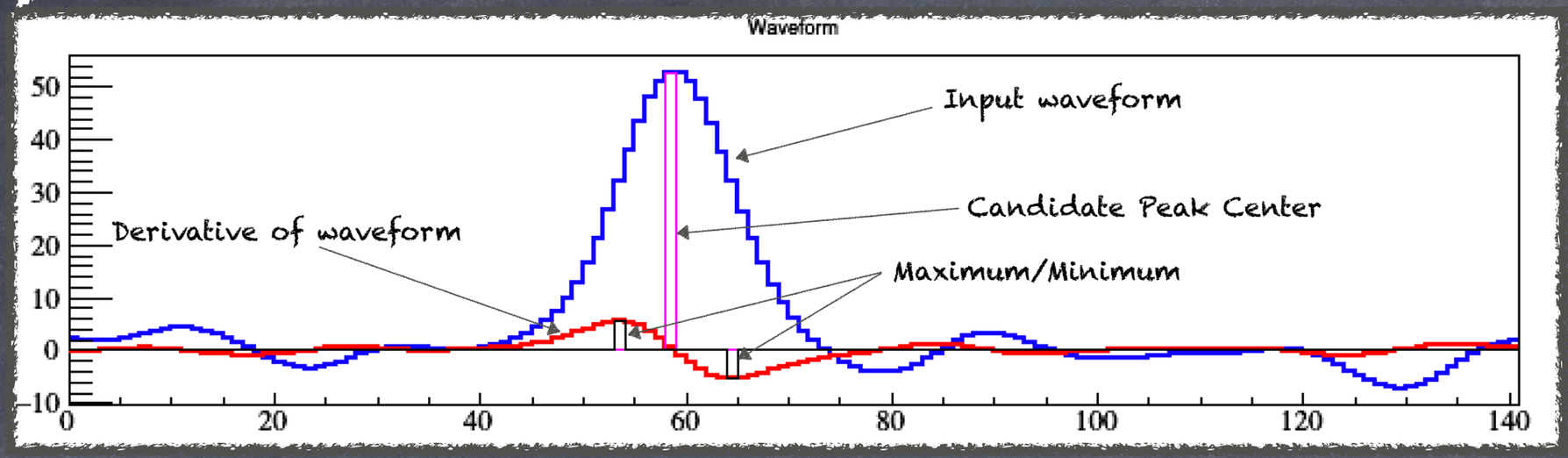
Standard Peak Finding



- Employs a "Bin over fixed threshold" approach
  - Assumes a zero baseline for waveform
  - Peak starts when bin goes over threshold, peak ends when bin drops below threshold
  - Returns candidate peak center and width to be used to seed gaussian fit
    - Peak center is max peak position, width are bins where threshold is crossed
- Simple, fast and works extremely well... but is not problem free
  - Can struggle to resolve very closely spaced hits (can lead to bad fits)
  - Can have problems when handed waveforms outside of basic assumption
    - For example baseline variations can be problematic
- Can we do better?

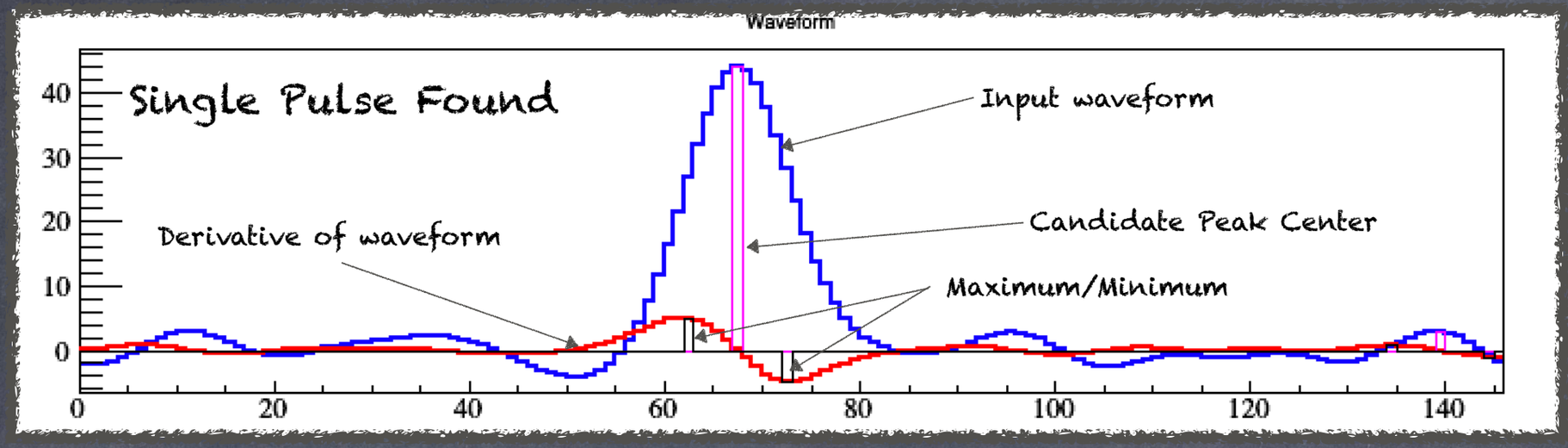
Note: default algorithm for GausHitFinder

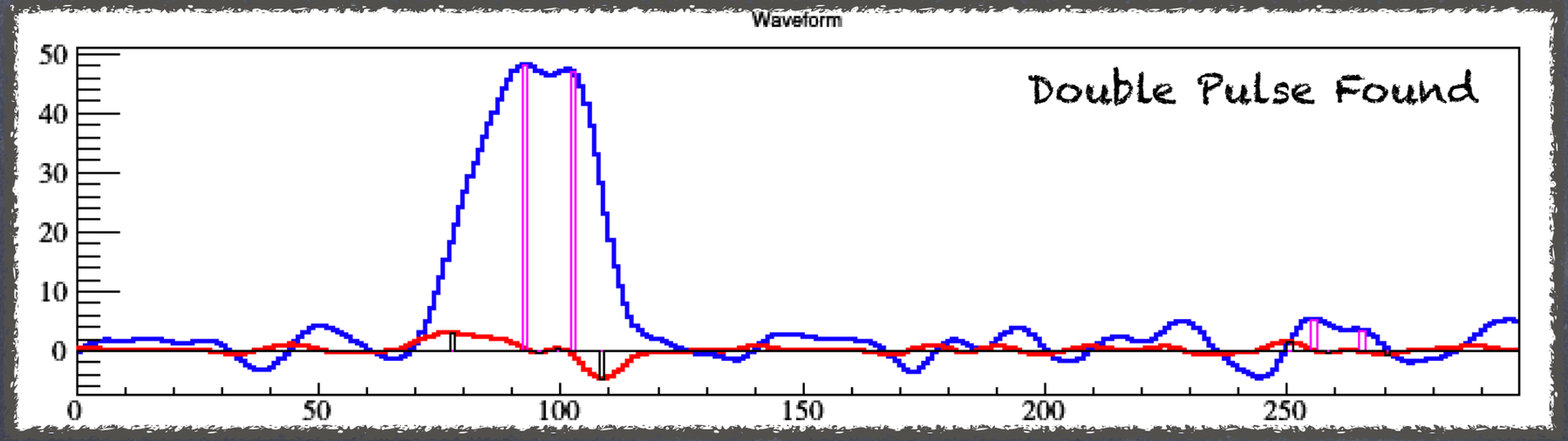
Differential Peak Finding

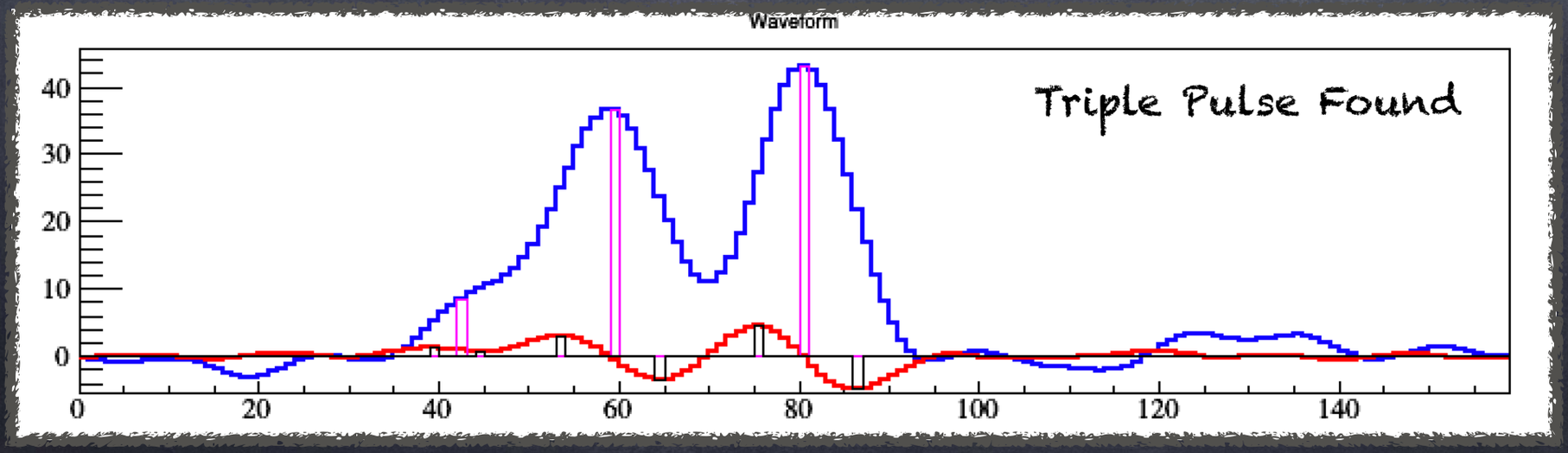


- Using a differential approach to finding hits reduces sensitivity to baseline movements and increases sensitivity to multiple peak separation
  - Form "smoothed" derivative of input waveform
  - Zero crossing defines candidate peak center
  - Maximum/minimum nearest peak center define width
  - Threshold now becomes difference from derivative maximum to minimum plus "range" between them

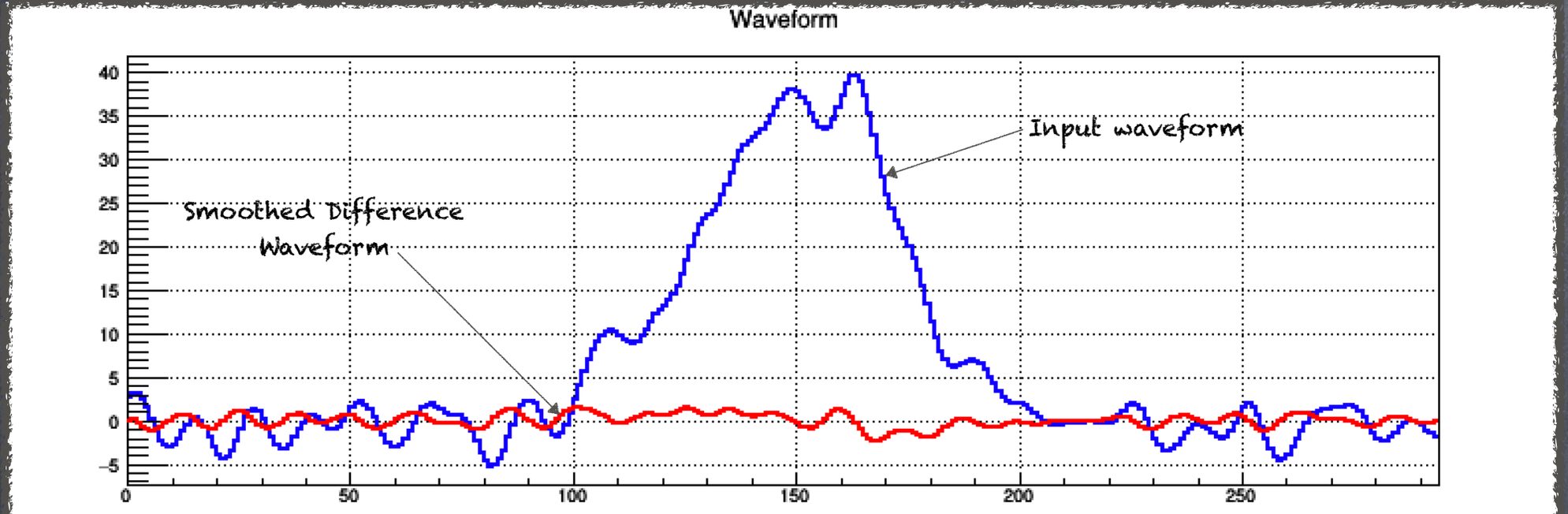
Candidate Peaks





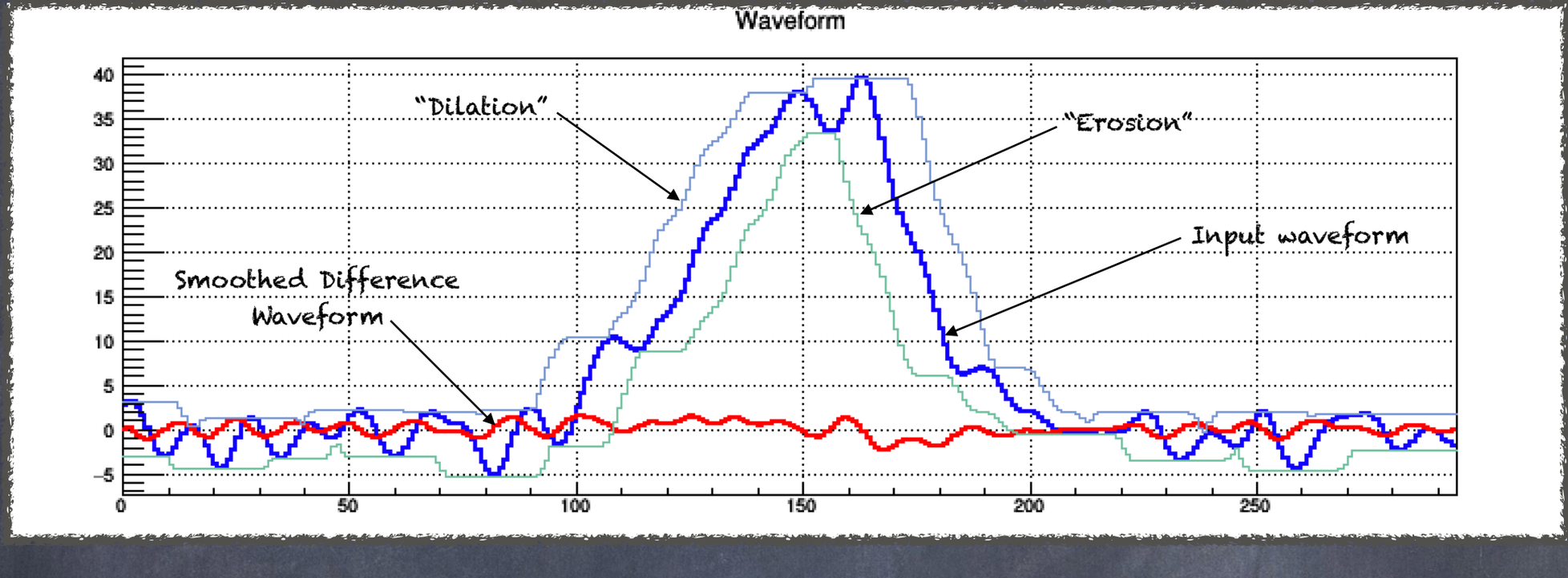


Differential Approach Failures



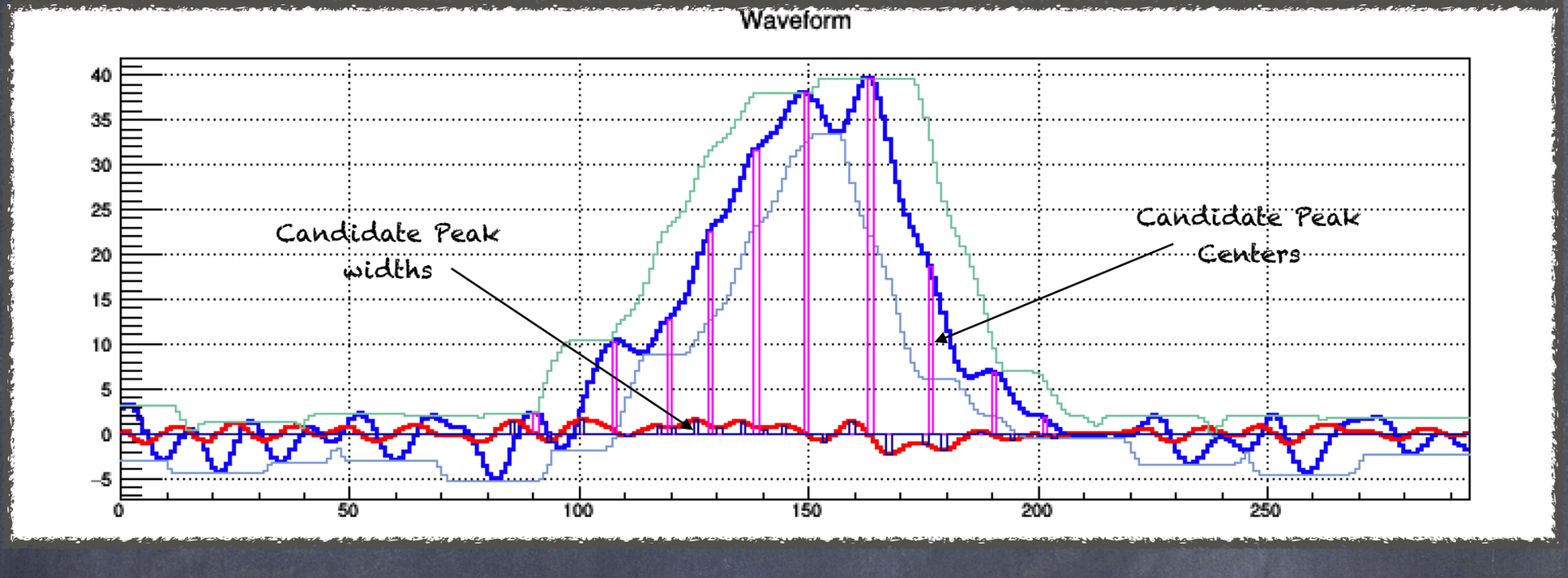
- Purely differential approach has its own issues:
  - There is still a threshold and it needs to be high enough to stay over the noise level...
  - Slowly rising/falling peaks will have correspondingly small derivatives which are not commensurate with the peak height
  - This can cause large pulses to disappear
- Can we do better?

Hybrid Peak Finding



- To regain sensitivity to slowly rising/falling waveforms employ a morphological filter to identify grouped peaks
  - Dilation and Erosion waveforms provide an "envelope" of the signal waveform - Look for large differences in the two
- Then use the differential method within a group to handle multiple hit separation

Hybrid Peak Finding



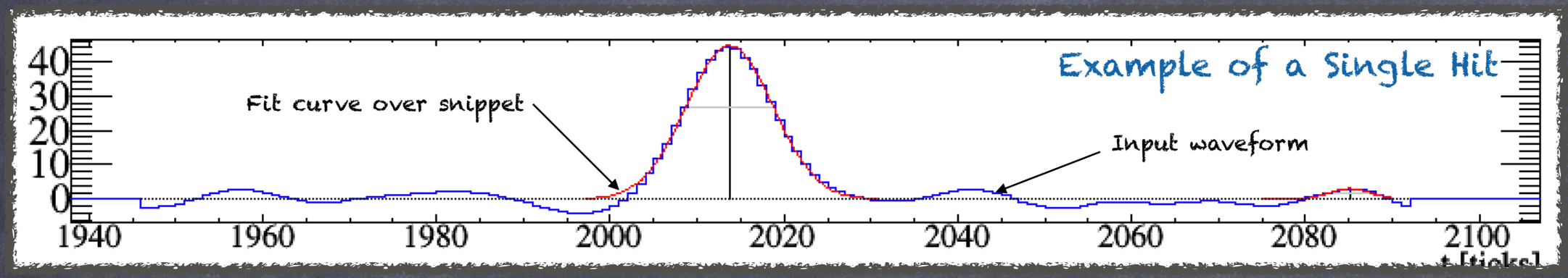
- To regain sensitivity to slowly rising/falling waveforms employ a morphological filter to identify grouped peaks
  - Provides "envelope" of the waveform
- Then use the differential method within a group to handle multiple hit separation
- This looks better!

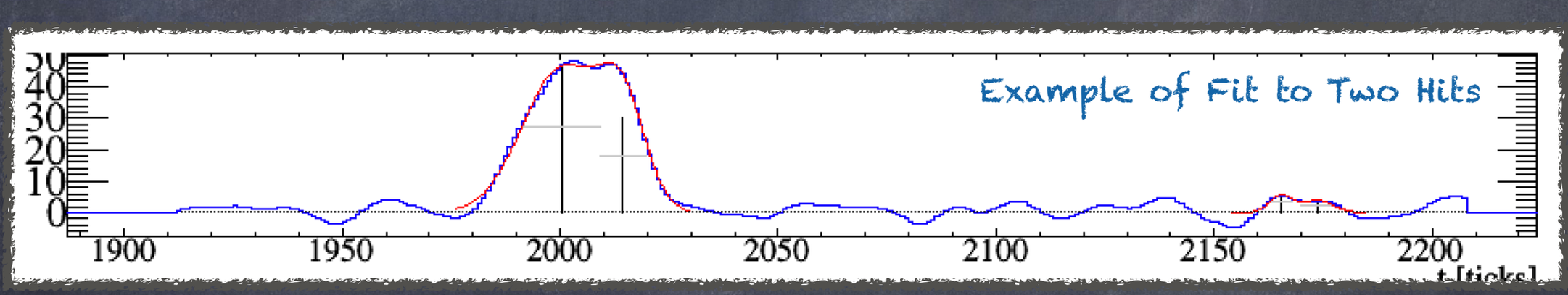
Peak Fitting
And
Special Case Handling

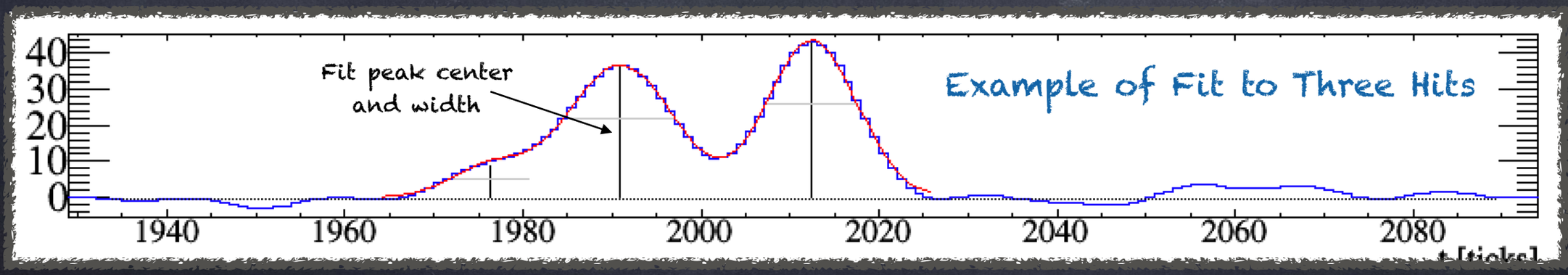
Peak Fitting

- Also implemented via a tool interface
- To date only one option available in LArReco
  - Operates on hit "snippets" which are a series of "adjacent" (i.e. touching) candidate hits
  - "Standard" fitting of n gaussian shapes to an input snippet
  - Initial parameters seeded by values from the candidate peak finding - guesses at peak center, amplitude and width
  - Worth noting:
    - Fitting is done with root: function defined by TF1 and uses TMinuit fitting on a histogram... Not really the optimal solution for events with large numbers of hits
- Definitely room for performance improvement here!

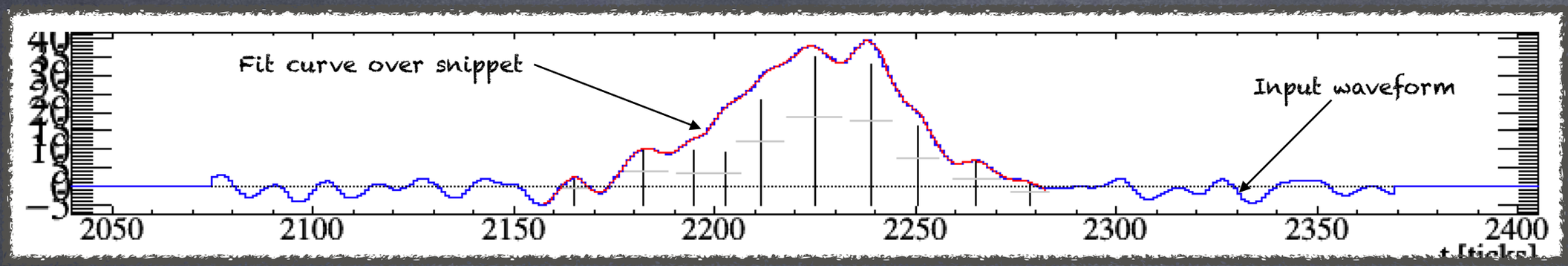
Examples of Fit Peaks

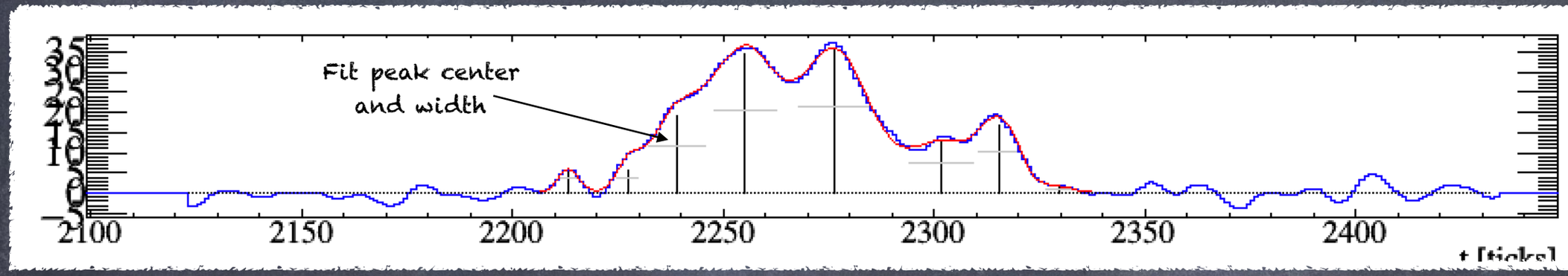






Hybrid Hit finder Fit Example



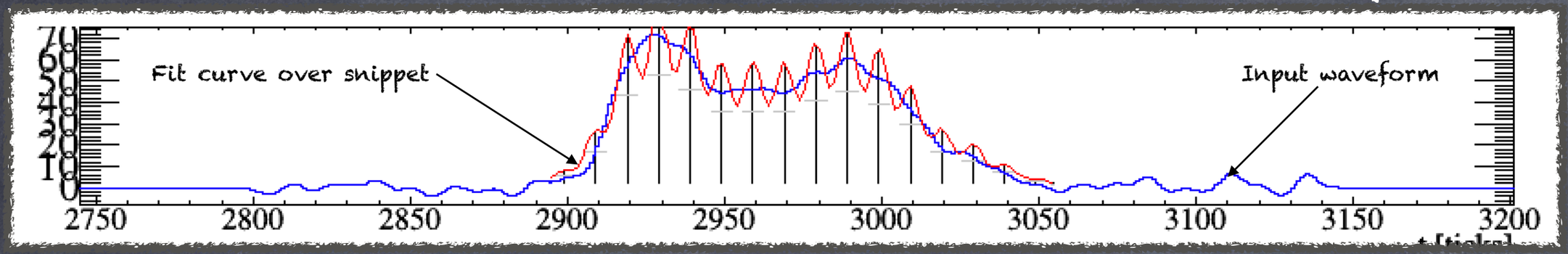


- Generally good fits can be made to the waveforms if all the candidate peaks can be found...
- First Problem: the more peaks the significantly larger amount of time taken to perform the fit...
- Second Problem: What does it mean to fit that many peaks?

Handling "Long" Waveforms

- It turns out that not all waveforms from LArTPC are a series of gaussian shaped pulses...
  - Particular problem are tracks that propagate very close to perpendicular to the wire planes
  - Produce long trains of deposited charge
- One can "fit" these with a large number of guassians
  - But what does this end up meaning?
  - As the number of candidate peaks grows, Minuit can begin to take a very significant amount of time for the fit to converge

"Long" Waveform Example



- Strategy for long waveforms (and fit failures) is to simply divide the waveform into a series of n gaussians
  - n is determined by dividing the length of the waveform by a $width_{min}$
  - -or- if this would result in too many hits then the waveform is broken into n_{max} gaussians of width snippet length / n_{max}
  - Area of gaussian take from ADC counts within 2*width
  - Number of hits to break into long waveform, n_{max}, $width_{max}$ are all fhicl parameters
- This is an area of the gauss hit finder that could use work

Controlling the Hit Finding

Top Level FHICL File

(see LArReco/HitFinder/hitfindermodules.fcl)

```
gaus_hitfinder:
{
   module_type:          "GausHitFinder"
   CalDataModuleLabel:   "caldata"
   MaxMultiHit:          10                # maximum hits for multi gaussia fit attempt
   AreaMethod:           0                 # 0 = area by integral, 1 = area by gaussian area formula
   AreaNorms:            [ 13.25, 26.31 ]  # normalizations that put signal area in
                                           # same scale as peak height.
   LongMaxHits:          [  1,  1,  1]     # max number hits in long pulse trains
   LongPulseWidth:       [ 16, 16, 16]     # max widths for hits in long pulse trains
   Chi2NDF:              2000              # maximum Chisquared / NDF allowed to store fit, if fail
                                           # will use "long" pulse method to return hit
   AllHitsInstanceName:  ""                # If non-null then this will be the instance name of all hits output to event
                                           # in this case there will be two hit collections, one filtered and one containing all hits

   # Candididate peak finding done by tool, one tool instantiated per plane (but could be other divisions too)
   HitFinderToolVec:
   {
      CandidateHitsPlane0:  @local::candhitfinder_standard  # plane 0
      CandidateHitsPlane1:  @local::candhitfinder_standard  # plane 1
      CandidateHitsPlane2:  @local::candhitfinder_standard  # plane 2
   }

   # Declare the peak fitting tool
   PeakFitter:           @local::peakfitter_gaussian

   # The below are for the hit filtering section of the gaushit finder
   FilterHits:           false             # true = do not keep undesired hits according to settings of HitFilterAlg object
   HitFilterAlg:
   {
      AlgName: "HitFilterAlg"
      MinPulseHeight: [5.0, 5.0, 5.0]     #minimum hit peak amplitude per plane
      MinPulseSigma:  [1.0, 1.0, 1.0]     #minimum hit rms per plane
   }
                                          # In addition to the filter alg we can also filter hits on the same pulse train
   PulseHeightCuts:      [3.0,  3.0,  3.0 ] # Minimum pulse height
   PulseWidthCuts:       [2.0,  1.5,  1.0 ] # Minimum pulse width
   PulseRatioCuts:       [0.35, 0.40, 0.20] # Ratio of pulse height to width
}
```

Below this box can be ignored

Tool Level FHICL File

(See LArReco/HitFinder/HitFinderTools)

```
#
# Define the differential tool for finding candidate hits
candhitfinder_derivative:
{
    tool_type:           CandHitDerivative
    Plane:               0
    MinDeltaTicks:       1
    MaxDeltaTicks:       40
    MinDeltaPeaks:       0.25
    MinHitHeight:        2
    NumInterveningTicks: 6
    OutputHistograms:    false
    WaveformAlgs:        @local::hitfinderwaveformalgs
}

# Define the morphological filter tool for finding candidate hits
candhitfinder_morphological:
{
    tool_type:           CandHitMorphological
    Plane:               0
    DilationThreshold:   4.
    DilationFraction:    0.75
    ErosionFraction:     0.2
    MinDeltaTicks:       1
    MinDeltaPeaks:       0.01
    MinHitHeight:        1
    NumInterveningTicks: 6
    StructuringElement:  20
    OutputHistograms:    false
    WaveformAlgs:        @local::hitfinderwaveformalgs
}

# Define the standard "threshold over baseline" method for finding candidate hits
candhitfinder_standard:
{
    tool_type:           CandHitStandard
    Plane:               0              # The plane this tool is operating on
    RoiThreshold:        5.             # The threshold to apply to find hits
}

peakfitter_gaussian:
{
    tool_type:      "PeakFitterGaussian"
    MinWidth:       0.5
    MaxWidthMult:   3.
    PeakRangeFact:  2.
    PeakAmpRange:   2.
    FloatBaseline:  false
}
```

The Standard hit finding tool now has only one fhicl parameter: the threshold for declaring a hit candidate

Note that these parameters represent limits, they are not the initial parameter estimates used by the fitter

Overriding Parameters

- FHICL overrides depend on the level one wants to have changed
  - May seem a trivial point but often see FHICL files where parameters are not being changed because of the use of tools...

- An example overriding parameters in both the module and the hit finding tool:

physics.producers.gaushit.CalDataModuleLabel: “recowire"
physics.producers.gaushit.LongMaxHits: [25, 25, 25]
physics.producers.gaushit.MaxMultiHit: 4
physics.producers.gaushit.Chi2NDF: 50.

physcis.producers.gaushit.HitFinderToolVec.CandidateHitsPlane0.RoiThreshold: 3.5
physics.producers.gaushit.HitFinderToolVec.CandidateHitsPlane1.RoiThreshold: 3.5
physics.producers.gaushit.HitFinderToolVec.CandidateHitsPlane2.RoiThreshold: 2.0

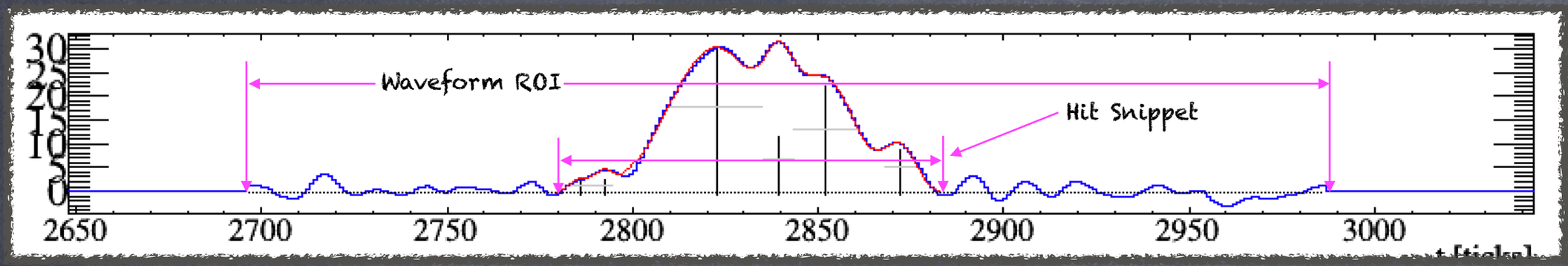
Output

recob::Hit Data Members

```
55
56      raw::ChannelID_t        fChannel;         ///< ID of the readout channel the hit was extracted from
57      raw::TDCtick_t          fStartTick;       ///< initial tdc tick for hit
58      raw::TDCtick_t          fEndTick;         ///< final tdc tick for hit
59      float                   fPeakTime;        ///< time of the signal peak, in tick units
60      float                   fSigmaPeakTime;   ///< uncertainty for the signal peak, in tick units
61      float                   fRMS;             ///< RMS of the hit shape, in tick units
62      float                   fPeakAmplitude;   ///< the estimated amplitude of the hit at its peak, in ADC units
63      float                   fSigmaPeakAmplitude; ///< uncertainty on estimated amplitude of the hit at its peak, in ADC units
64      float                   fSummedADC;       ///< the sum of calibrated ADC counts of the hit
65      float                   fIntegral;        ///< the integral under the calibrated signal waveform of the hit, in tick x ADC units
66      float                   fSigmaIntegral;   ///< the uncertainty of integral under the calibrated signal waveform of the hit, in ADC units
67      short int               fMultiplicity;    ///< how many hits could this one be shared with
68      short int               fLocalIndex;      ///< index of this hit among the Multiplicity() hits in the signal window
69      float                   fGoodnessOfFit;   ///< how well do we believe we know this hit?
70      int                     fNDF;             ///< degrees of freedom in the determination of the hit shape
71      geo::View_t             fView;            ///< view for the plane of the hit
72      geo::SigType_t          fSignalType;      ///< signal type for the plane of the hit
73      geo::WireID             fWireID;          ///< WireID for the hit (Cryostat, TPC, Plane, Wire)
74
```

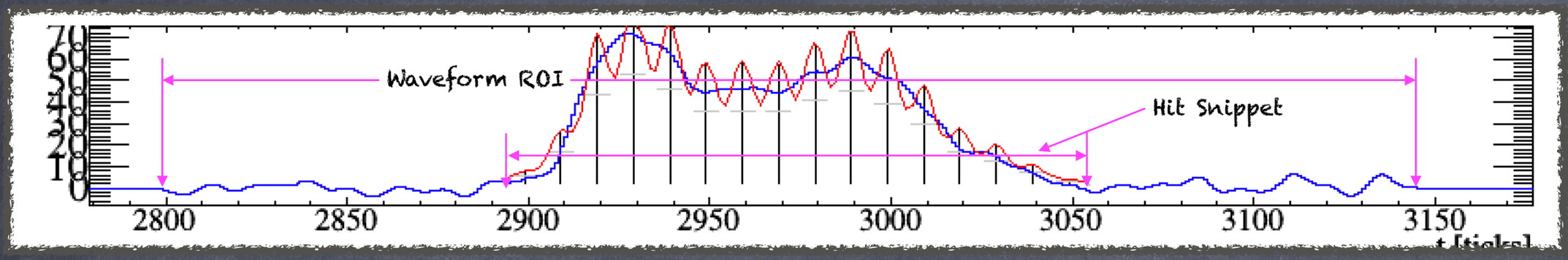
- Of note:
  - Hits are contained on "snippets" which are subsets of the recon::Wire ROI given to hit finder. The "StartTick" and "EndTick" are the endpoints of the snippet
  - Multiplicity is the number of hits on the snippet
  - LocalIndex tells you which hit in the multiplet

Features of the Output



- Two methods for returning the estimated charge of the hit:
  - "Integral": Area of the gaussian determined by the fit to the waveform
  - "SummedADC": Sum of ADC values from the start to the end of the hit snippet
- For single hits the two methods should give the same result
- For Multiple hits the SummedADC ~ Sum of the Integrals of the individual hits

Features of the Output



- If the Fit χ^2 is bad:
  - NDF will be set to 1
  - χ^2 will be the "bad" value
- If the number of candidate hits exceeds the long waveform selection cut:
  - NDF will be set to 1
  - χ^2 will be set to -1

Where Next?

The Future of GausHitFinder

- Short Range Improvements
  - Can we spell "Gauss" correctly?
  - Better documentation of the tool FHICL parameters
  - Remove hit filtering - Experiment specific, should be independent module?
  - Break handling of long waveforms out to an art tool
- HPC
  - e.g. PNNL has proof of principle for vectorization?
- Candidate hit finding in 2D?
  - Morphological filtering comes from 2D image reconstruction, is straightforward to apply to hit finding problem
    - Here would need a new top level module since GausHitFinder architected to loop over wires
  - This may have more application as a 2D ROI finder to facilitate "sparse RawDigits" to try to reduce data volume - but off topic here...
- Other ideas?