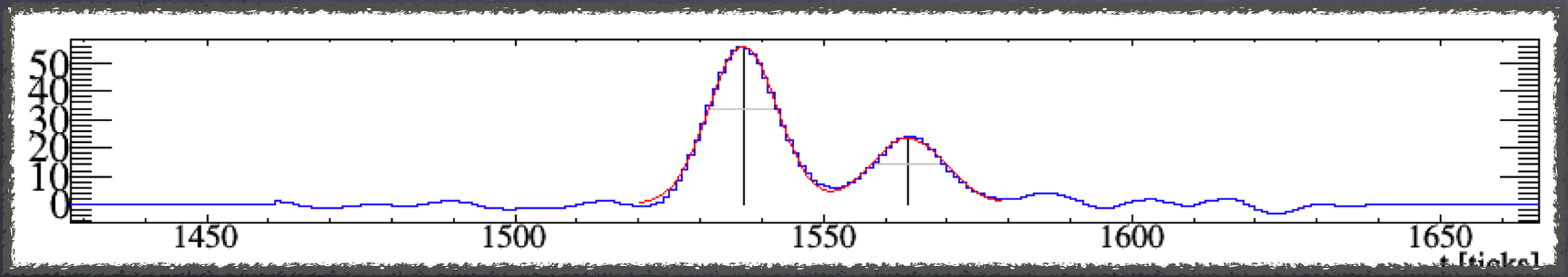


GausHit Finder Updates

Tracy Usher

LArSoft Coordination Meeting

June 19, 2018

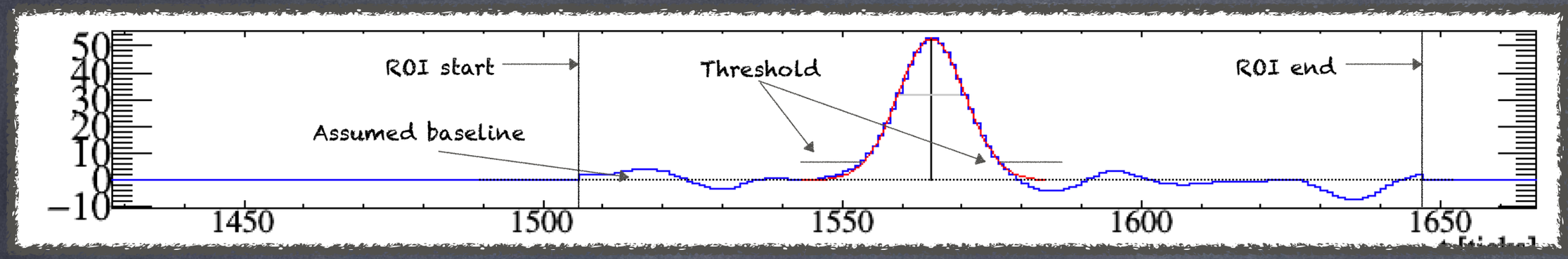
The Short Version

- Requesting to make small update to gaushit finder to facilitate work in ICARUS
  - Would like ability to switch in different candidate peak fitting "tools" - possibly plane dependent
  - Slight change to interface to peak finding tools - probably no impact to outside users - not breaking
  - Small change to fhicl definition - "breaking" change for users of gaushit finder to accommodate fhicl update
    - I believe these changes would all occur in larreco
  - Standard users of gaushit (e.g. MicroBooNE) should not see a change in output of module

Start Here: The Longer Version

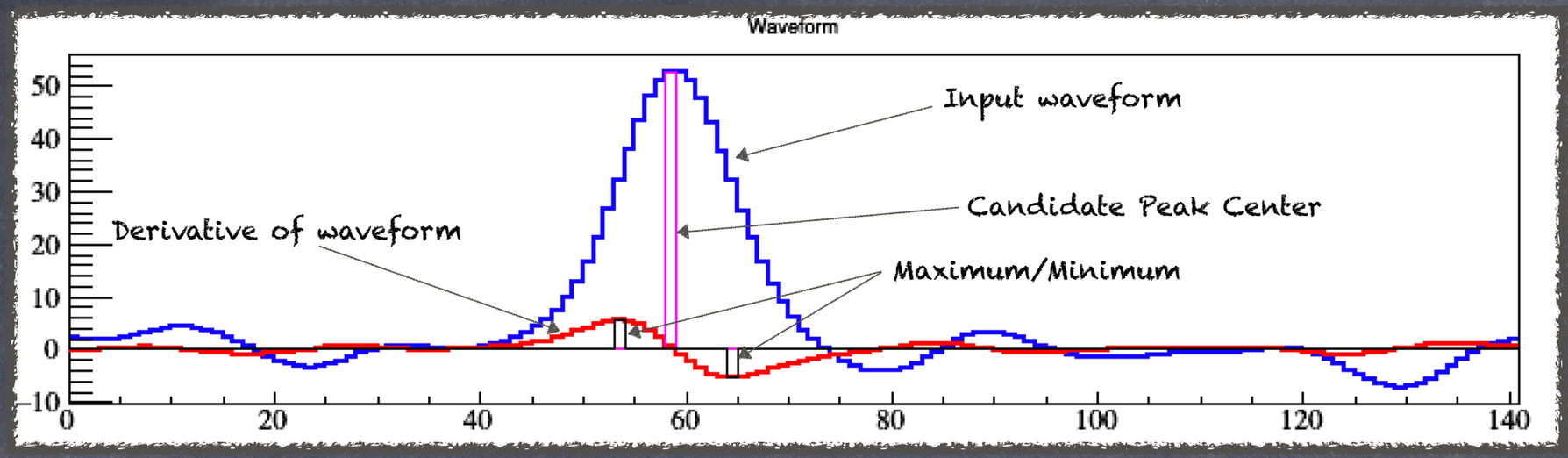
- Gaushit finder was refactored some time ago:
  - Separate "candidate hit finding" from "hit fitting"
  - Both implemented with art tools through interface classes
    - ICanHitFinder.h for tools finding candidate peaks
    - IPeakFitter.h for tools to fit the peaks
- Note: in addition to the tools available in larreco (see larreco/HitFinder/HitFinderTools) this interface allows the ability for experiments to implement their own versions of these tools

Gaushit Standard Peak Finding



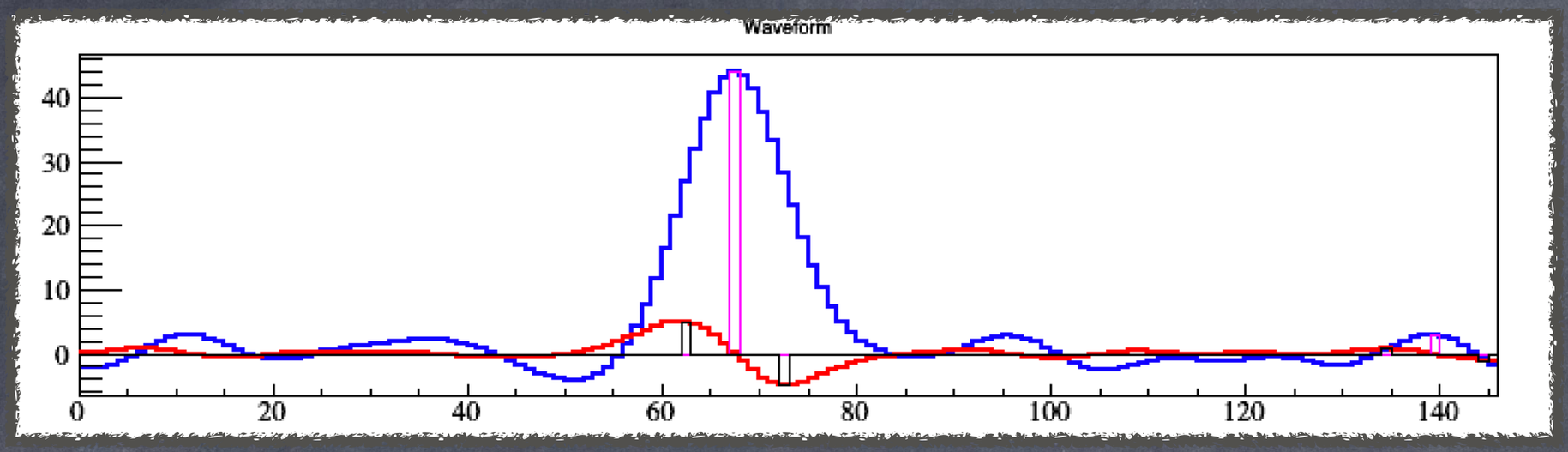
- Used a "Bin over fixed threshold" approach
  - Assumed a zero baseline for waveform
    - Peak starts when bin goes over threshold, peak ends when bin drops below threshold
    - If peak end - start large enough then return candidate pulse
      - Candidate peak center and width used to seed a gaussian fit of waveform
- Works fine in bulk of the cases with single and even multiple hits
  - But has a number of well known failures for "special" cases... and where this is a significant enough problem to want to do something better...
    - For example baseline variations can be problematic

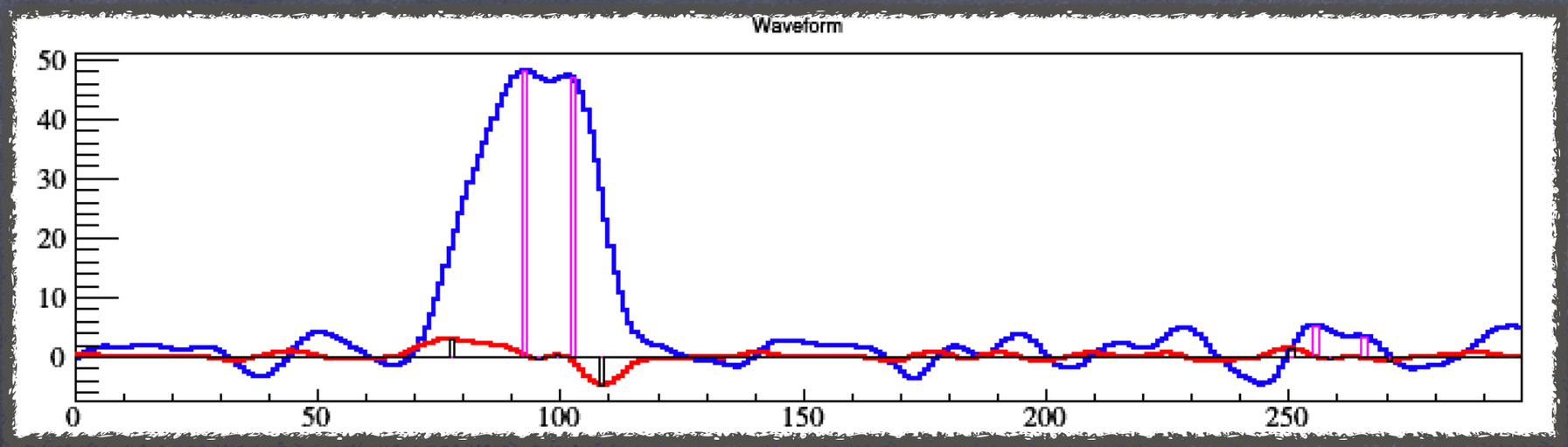
Gaushit Alternate Peak Finding

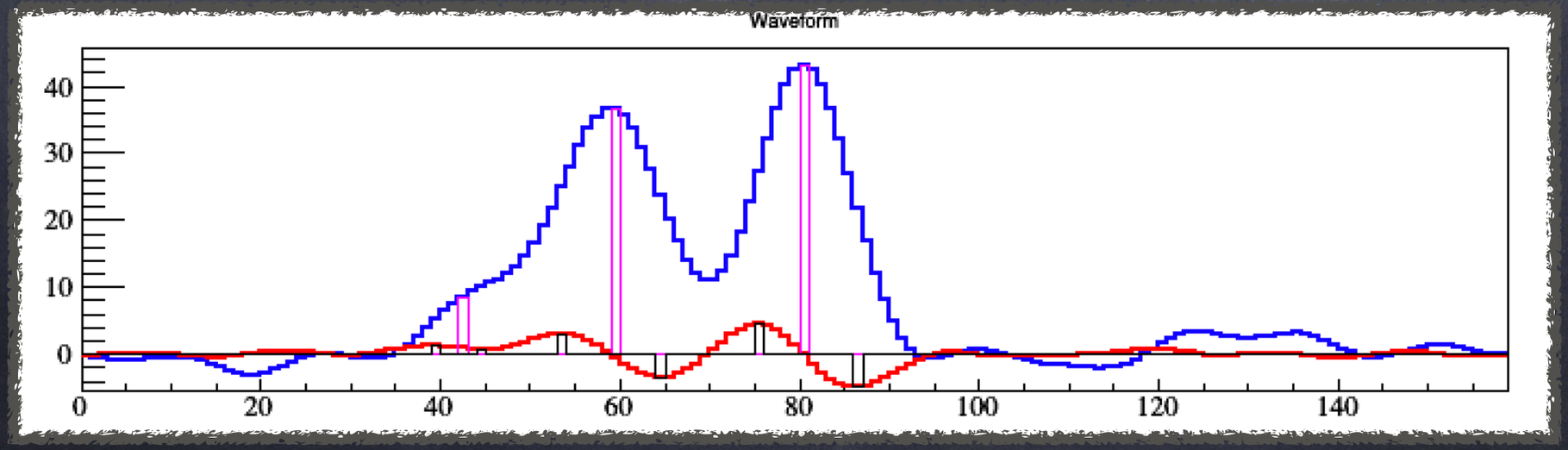


- Use a differential approach to reduce sensitivity to baseline movements and increase sensitivity to multiple peak separation
  - Form "smoothed" derivative
  - Zero crossing defines candidate peak center
  - Maximum/minimum nearest peak center define width
  - Threshold now becomes difference from derivative maximum to minimum plus "range" between them

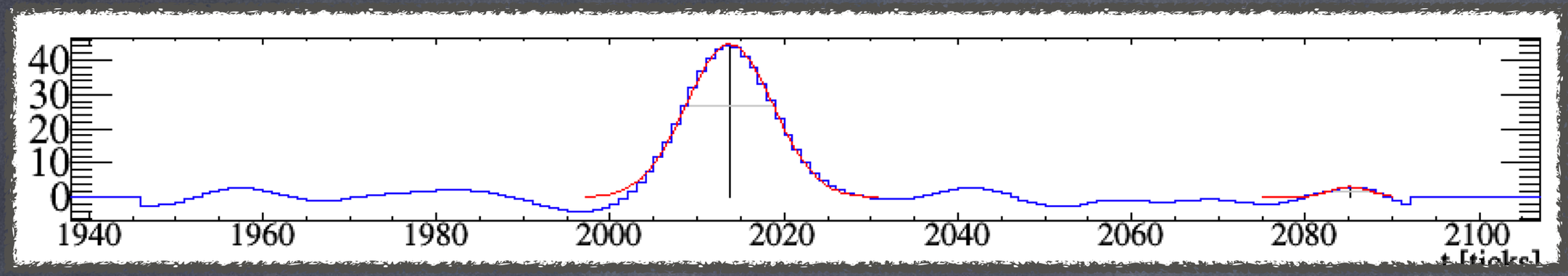
Candidate Peaks

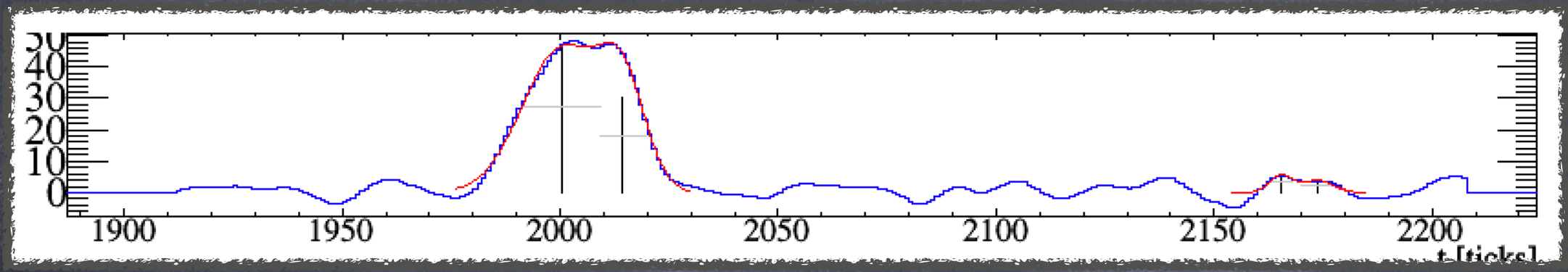


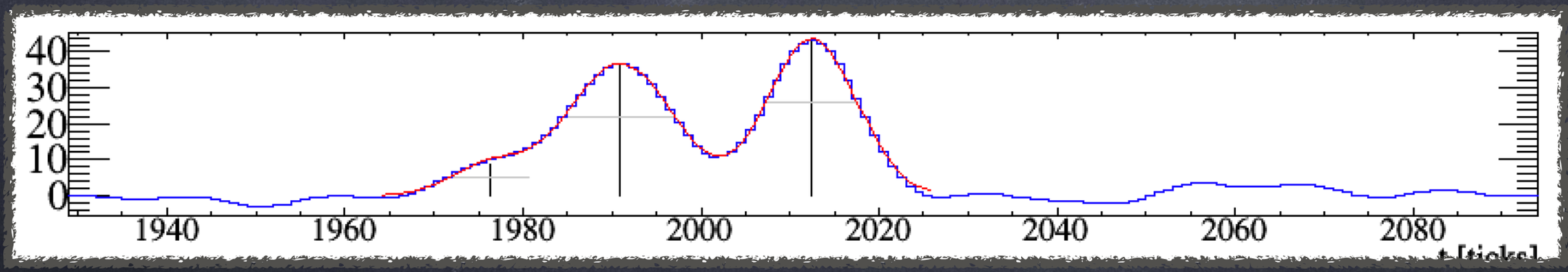




Fit Peaks


40
30
20
10
0
1940
1960
1980
2000
2020
2040
2060
2080
2100


40
30
20
10
0
1900
1950
2000
2050
2100
2150
2200


40
30
20
10
0
1940
1960
1980
2000
2020
2040
2060
2080

A Quick Word On Peak Fitting

- Also implemented via a tool interface
- To date only one option available
  - "Standard" fitting of n gaussian shapes to an input waveform
  - Initial parameters seeded by values from the candidate peak finding
  - Uses TMinuit fitting on a histogram... not the most efficient procedure particularly when processing several thousand hits per event

Gaushit General Status

- Almost all experiments (using gaushit) use the threshold over fixed baseline algorithm
  - The differential approach still needs work for better handling of very long waveforms but is being actively developed for ICARUS
- The current peak fitting could be improved
  - Move away from TMinuit/histograms?
  - Allow for more complex fitting functions (e.g. skew gaussian)
- There remains more work to be done to better understand the handling of long waveforms
  - These clearly break from the assumption of a guassian shaped hit
  - Is there clear consensus on the "right" way to handle these?
- The gaushit finder remains a work in progress!

Update 1: More Flexibility

- Tool interface to candidate hit finding and fitting allows for experiment specific implementations
  - This might want to be plane dependent
- Current interface is one tool for all candidate hit finding
  - Thresholds, etc., are stored in a vector referenced by ... (used to be view, now plane)
- It is desirable to allow a tool per plane to be used
  - Code changes are straightforward and, as said, are not by themselves breaking
  - Requires a modification to the fhicl definitions and this would be breaking
- In any case the current users of gaushit should not see any differences in the output of the module, only in how it is set up.

Update 2: Candidate Hit Interface

- The interface definition for the candidate hit finder tool could use some updates and cleanup
  - Would like to include the ability to make diagnostic histograms for debugging
    - Fhicl controllable to turn off for production
- Take opportunity to clean up
  - In particular only one clean entry point for candidate hit finding, the second definition is meant to be for internal use only

Proposed Updates

- Modifications described above to change the interface to hit finding and to allow for an independent tool per plane
  - Requires an update to user fhicl files
  - Should not result in output changes for those users obtaining hits from gaushit via the standard method of threshold over baseline
- Changes are included on larreco feature branch: feature/usher_cluster3dupdates
  - Note: in conjunction with similarly named feature branch in lardata also brings in updates to Cluster3D which impacts no external users
    - Cleans up a mess with object defintions in lardata,
    - Significant updates in the method which can be available to satisfy some outstanding requests (e.g. the deep learn physics group)

Differences

Current FHICL Definition

```
35
36 gaus_hitfinder:
37 {
38  module_type:          "GausHitFinder"
39  CalDataModuleLabel:   "caldata"
40  MinSig:               [ 6.0, 11.0 ]      # signal height threshold, per plane
41  MinWidth:             [ 4.0, 6.0 ]       # hit width threshold, per plane
42  MaxMultiHit:          10                 # maximum hits for multi fit
43  AreaMethod:           0                  # 0 = area by integral, 1 = area by gaussian area formula
44  AreaNorms:            [ 13.25, 26.31 ]   # normalizations that put signal area in
45                                           # same scale as peak height.
46  LongMaxHits:          [  1,  1,  1]      # max number hits in long pulse trains
47  LongPulseWidth:       [ 16, 16, 16]      # max widths for hits in long pulse trains
48  TryNplus1Fits:        true                      # true will try to re-fit poorly modeled hits with n+1 gaussians
49                                                  # false will NOT try to re-fit poorly modled hits
50  Chi2NDF:              2000               # maximum Chisquared / NDF allowed for a hit to be saved (Set very high by default)
51
52  AllHitsInstanceName:  ""                 # If non-null then this will be the instance name of all hits output to event
53                                           # in this case there will be two hit collections, one filtered and one containing all hits
54  FilterHits:           false              # true = do not keep undesired hits according to settings of HitFilterAlg object
55  CandidateHits:        @local::candhitfinder_standard
56  PeakFitter:           @local::peakfitter_gaussian
57  HitFilterAlg:
58  {
59    AlgName: "HitFilterAlg"
60    MinPulseHeight: [5.0, 5.0, 5.0]        #minimum hit peak amplitude per plane
61    MinPulseSigma:  [1.0, 1.0, 1.0]        #minimum hit rms per plane
62  }
63                                           # In addition to the filter alg we can also filter hits on the same pulse train
64  PulseHeightCuts:      [3.0,  3.0,  3.0 ] # Minimum pulse height
65  PulseWidthCuts:       [2.0,  1.5,  1.0 ] # Minimum pulse width
66  PulseRatioCuts:       [0.35, 0.40, 0.20] # Ratio of pulse height to width
67 }
68
```

Proposed FHICL Definition

```
35
36 gaus_hitfinder:
37 {
38  module_type:          "GausHitFinder"
39  CalDataModuleLabel:   "caldata"
40  MinSig:               [ 6.0, 11.0 ]      # signal height threshold, per plane
41  MinWidth:             [ 4.0, 6.0 ]       # hit width threshold, per plane
42  MaxMultiHit:          10                 # maximum hits for multi fit
43  AreaMethod:           0                  # 0 = area by integral, 1 = area by gaussian area formula
44  AreaNorms:            [ 13.25, 26.31 ]   # normalizations that put signal area in
45                                           # same scale as peak height.
46  LongMaxHits:          [  1,  1,  1]      # max number hits in long pulse trains
47  LongPulseWidth:       [ 16, 16, 16]      # max widths for hits in long pulse trains
48  TryNplus1Fits:        true                      # true will try to re-fit poorly modeled hits with n+1 gaussians
49                                                  # false will NOT try to re-fit poorly modled hits
50  Chi2NDF:              2000               # maximum Chisquared / NDF allowed for a hit to be saved (Set very high by default)
51
52  AllHitsInstanceName:  ""                 # If non-null then this will be the instance name of all hits output to event
53                                           # in this case there will be two hit collections, one filtered and one containing all hits
54  FilterHits:           false              # true = do not keep undesired hits according to settings of HitFilterAlg object
55  PeakFitter:           @local::peakfitter_gaussian
56  HitFilterAlg:
57  {
58    AlgName: "HitFilterAlg"
59    MinPulseHeight: [5.0, 5.0, 5.0]        #minimum hit peak amplitude per plane
60    MinPulseSigma:  [1.0, 1.0, 1.0]        #minimum hit rms per plane
61  }
62  HitFinderToolVec:
63  {
64    CandidateHitsPlane0:  @local::candhitfinder_standard
65    CandidateHitsPlane1:  @local::candhitfinder_standard
66    CandidateHitsPlane2:  @local::candhitfinder_standard
67  }
68                                           # In addition to the filter alg we can also filter hits on the same pulse train
69  PulseHeightCuts:      [3.0,  3.0,  3.0 ] # Minimum pulse height
70  PulseWidthCuts:       [2.0,  1.5,  1.0 ] # Minimum pulse width
71  PulseRatioCuts:       [0.35, 0.40, 0.20] # Ratio of pulse height to width
72 }
73
74 gaus_hitfinder.HitFinderToolVec.CandidateHitsPlane0.Plane: 0
75 gaus_hitfinder.HitFinderToolVec.CandidateHitsPlane1.Plane: 1
76 gaus_hitfinder.HitFinderToolVec.CandidateHitsPlane2.Plane: 2
77
```

Legacy and should be removed at some point

Current Interface Class Def

```
class ICandidateHitFinder
{
public:
    virtual ~ICandidateHitFinder() noexcept = default;

    // Define standard art tool interface
    virtual void configure(const fhicl::ParameterSet& pset) = 0;

    // Define a structure to contain hits
    using HitCandidate_t = struct HitCandidate
    {
        size_t startTick;
        size_t stopTick;
        size_t maxTick;
        size_t minTick;
        float  maxDerivative;
        float  minDerivative;
        float  hitCenter;
        float  hitSigma;
        float  hitHeight;
    };

    using HitCandidateVec      = std::vector<HitCandidate_t>;
    using MergeHitCandidateVec = std::vector<HitCandidateVec>;

    using Waveform = std::vector<float>;

    // Search for candidate hits on the input waveform
    virtual void findHitCandidates(const Waveform&,                 // Waveform to analyze
                                   size_t,                          // waveform start tick
                                   double,                          // threshold
                                   HitCandidateVec&) const = 0;     // output candidate hits

    // Search for candidate hits on the input waveform
    virtual void findHitCandidates(Waveform::const_iterator,        // Start of waveform
                                   Waveform::const_iterator,        // end of waveform
                                   size_t,                          // waveform start tick
                                   double,                          // threshold
                                   HitCandidateVec&) const = 0;     // output candidate hits

    virtual void MergeHitCandidates(const Waveform&,
                                    const HitCandidateVec&,
                                    MergeHitCandidateVec&) const = 0;
};
```

Modify

Remove

Proposed Interface Class Def

```
class ICandidateHitFinder
{
public:
    virtual ~ICandidateHitFinder() noexcept = default;

    // Define standard art tool interface
    virtual void configure(const fhicl::ParameterSet& pset) = 0;

    // Define a structure to contain hits
    using HitCandidate_t = struct HitCandidate
    {
        size_t startTick;
        size_t stopTick;
        size_t maxTick;
        size_t minTick;
        float  maxDerivative;
        float  minDerivative;
        float  hitCenter;
        float  hitSigma;
        float  hitHeight;
    };

    using HitCandidateVec      = std::vector<HitCandidate_t>;
    using MergeHitCandidateVec = std::vector<HitCandidateVec>;

    using Waveform = std::vector<float>;

    // Search for candidate hits on the input waveform
    virtual void findHitCandidates(const Waveform&,                    // Waveform to analyze
                                   size_t,                             // waveform start tick
                                   size_t,                             // channel #
                                   size_t,                             // Event count (for histograms)
                                   HitCandidateVec&) const = 0;        // output candidate hits

    virtual void MergeHitCandidates(const Waveform&,
                                    const HitCandidateVec&,
                                    MergeHitCandidateVec&) const = 0;
};
```

Could imagine generalizing the hit candidate structure in the future