Timing conventions for recob waveforms

Chris Backhouse (UCL)

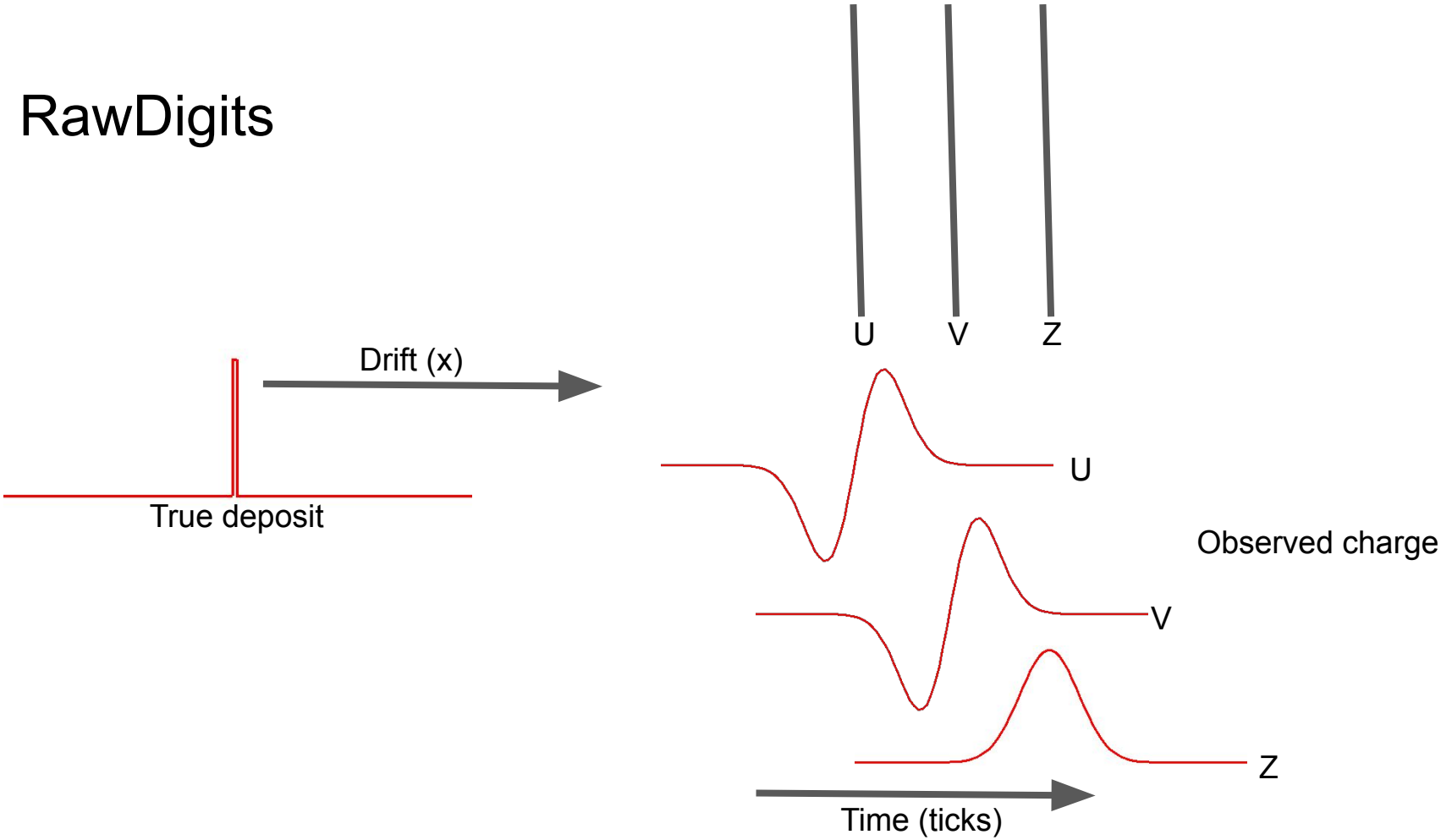
LArSoft coordination meeting

16 Nov 2021

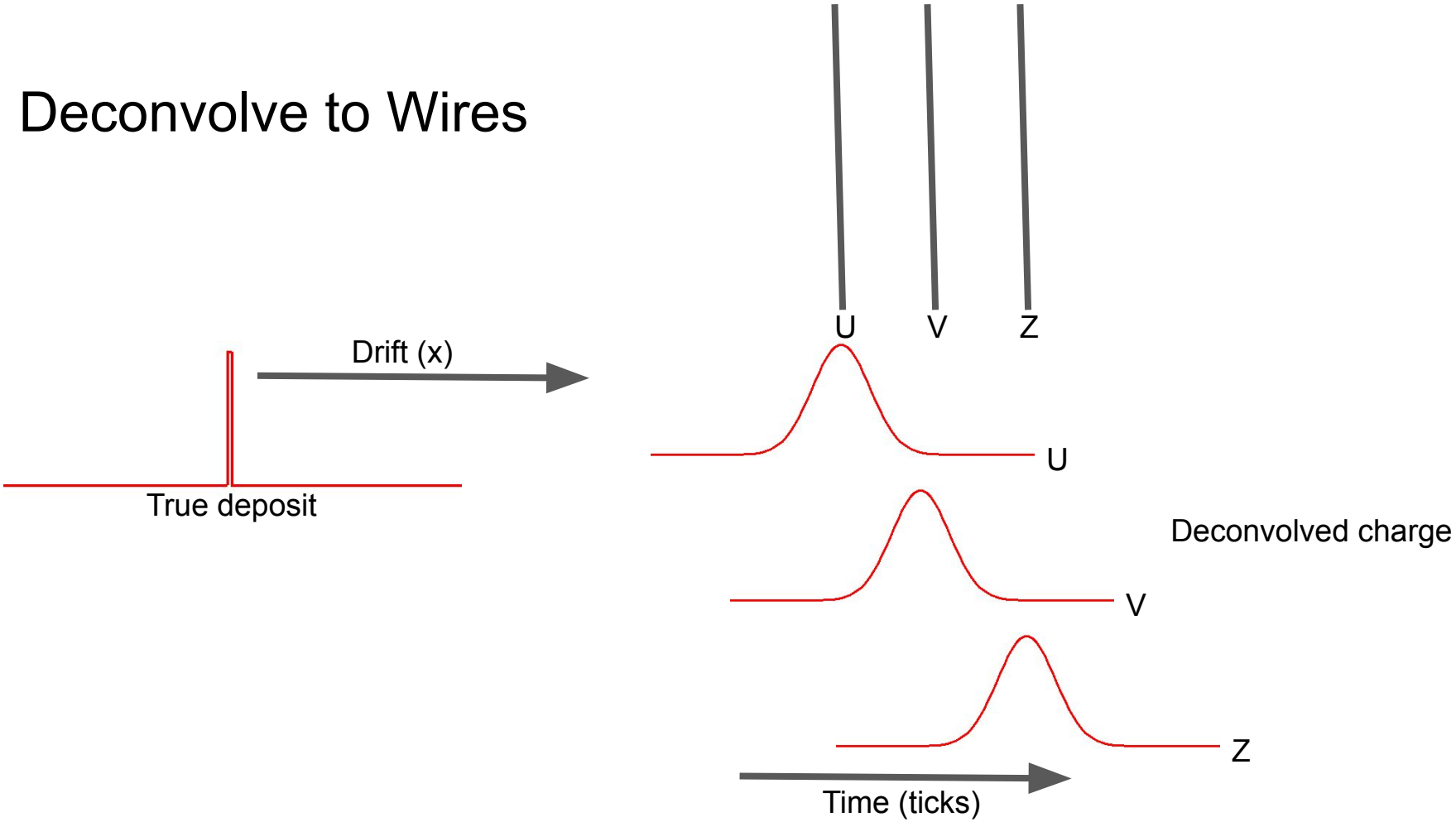
Introduction

- Preparing for new DUNE FD production we were seeing broken reconstruction
- Took a long time to track the problem down, because it’s relatively subtle
- This talk is a PSA to other experiments
- Also a request to document conventions and improve the interfaces to make it harder to do the wrong thing
- Hope to spark a discussion

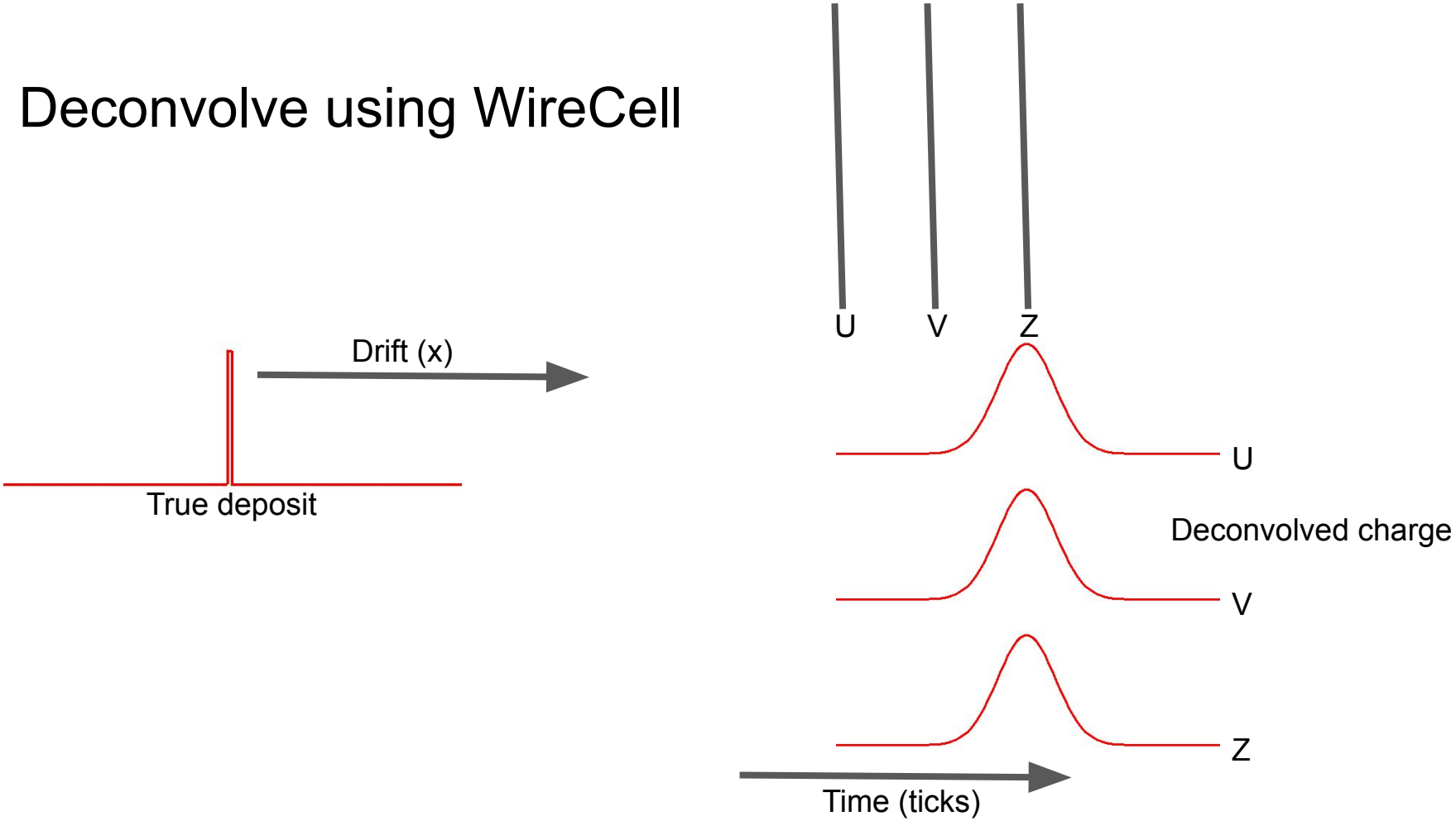
RawDigits



Deconvolve to Wires



Deconvolve using WireCell



Impact on reconstruction

- Reconstruction is critically dependent on matching charge deposits by time between views
- At least WebEVD and SpacePointSolver use handy ConvertTicksToX() function
- I believe Pandora does too

DetectorPropertiesData

double	**ConvertXToTicks** (double X, int p, int t, int c) const
double	**ConvertXToTicks** (double X, **geo::PlaneID** const &planeid) const
double	**ConvertTicksToX** (double ticks, int p, int t, int c) const
double	**ConvertTicksToX** (double ticks, **geo::PlaneID** const &planeid) const
double	**GetXTicksOffset** (int p, int t, int c) const
double	**GetXTicksOffset** (**geo::PlaneID** const &planeid) const
double	**GetXTicksCoefficient** (int t, int c) const
double	**GetXTicksCoefficient** (**geo::TPCID** const &tpcid) const
double	**GetXTicksCoefficient** () const

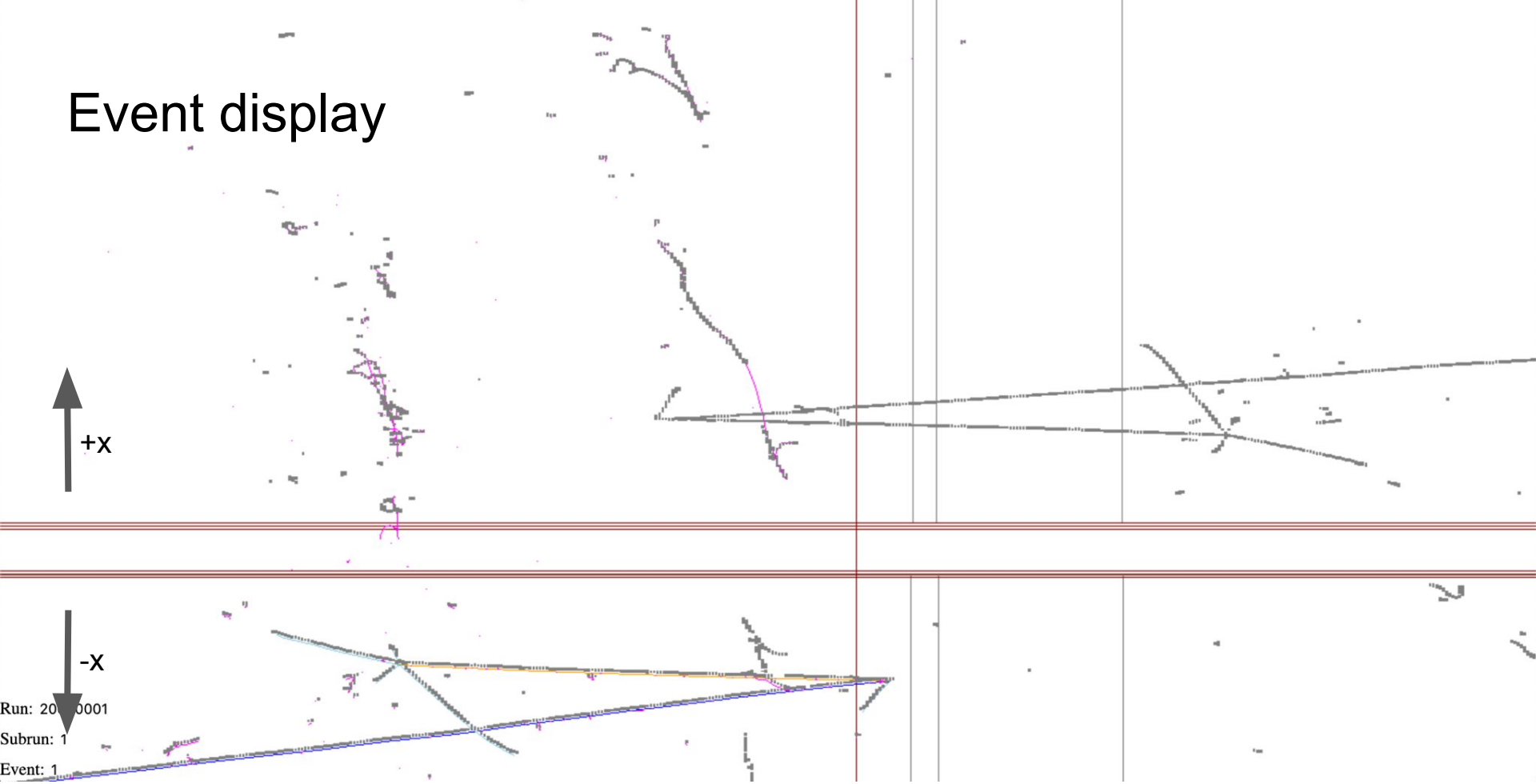
Impact on reconstruction

- Reconstruction is critically dependent on matching charge deposits by time between views
- At least WebEVD and SpacePointSolver use handy ConvertTicksToX() function
- I believe Pandora does too
- RawDigits and Wires both have ticks+planeID, but the meaning is *subtly* different

DetectorPropertiesData

double	**ConvertXToTicks** (double X, int p, int t, int c) const
double	**ConvertXToTicks** (double X, **geo::PlaneID** const &planeid) const
double	**ConvertTicksToX** (double ticks, int p, int t, int c) const
double	**ConvertTicksToX** (double ticks, **geo::PlaneID** const &planeid) const
double	**GetXTicksOffset** (int p, int t, int c) const
double	**GetXTicksOffset** (**geo::PlaneID** const &planeid) const
double	**GetXTicksCoefficient** (int t, int c) const
double	**GetXTicksCoefficient** (**geo::TPCID** const &tpcid) const
double	**GetXTicksCoefficient** () const

Event display



Discussion I

- ProtoDUNE-SP already “solved” this problem by running with a version of DetectorProperties that doesn’t (double-)correct the timing offsets
- We are likely to do this for upcoming FD production
- Not a great solution: now no way to decode RawDigits, or use old deconvolution methods
- Ought to audit all code making such conversions, some may already have worked around problems internally, or applied unnecessarily large timing tolerances
- Suspect other experiments may have a patchwork of workarounds

Discussion II

- Are these the conventions that were intended / that we want? Should document
- Can we improve the interface to be harder to misuse?
  - Only take concrete RawDigit or Wire objects?
  - Use different units/offset in Wire so mistakes are obvious?
- WireCell philosophy is to recover true charge distribution, close analogy to RawDigits unnecessary/unhelpful
- Had to assume a field strength to do this. At that point, why not directly express Wires in terms of distance to readout, rather than electronics-level ticks?

- For wrapped wires this all breaks down if the spacing or field differs in the two TPCs. Theoretical-only concern, or sign of real confusion?

Other comments about conventions

- **Tracy**: Icarus moving from 1D deconvolution to WireCell
- Problems with conflation of concepts of view vs plane. Does current service make this error?
  - Not sure what we can do about this except be on the lookout for it
  - Would using *enum class* for plane and view help?
- Would like to set lifetime by TPC, or at least cryostat

- **Brett**: VD coldbox also has Icarus-like geometry
- Is there still an arbitrary scale factor applied to make Wires “ADC sized”?
  - Personally, I agree they should be “electron sized”

Backup

We recently encountered an issue in DUNE related to inconsistencies in the conventions for timing between RawDigits, Wires, the DetectorProperties service, and various reconstruction algorithms. We believe we now understand the problem, and are able to make progress, but the fix isn't the best and points to some deeper philosophical issues.

RawDigits encode recorded waveforms on a wire. The timing is fairly uncontroversial, though it may be tricky to allow for various global offsets in practice.

A recob::Wire is the deconvolved version of a RawDigit, and is intended to represent something closer to the true charge incident on the detector readout. In most (all?) experiments this conversion is done by WireCell.

The DetectorProperties service provides a useful ConvertTicksToX() function. Reconstruction algorithms may make use of this function. One reason to do so is to perform matching of hits between views. If they map to a similar X value, they are likely related. Our investigations focused on SpacePointSolver, but I think this is a generic thing many algorithms do.

Due to the small distance between the wire readout planes in different views, a single charge deposit will produce slightly earlier signals in the induction wires than in the collection wires. The DUNE FD DetectorProperties service was configured to remove these offsets so that, e.g., converting RawDigits from different views to X using this function would lead to a visually pleasing overlap between the signals.

It turns out that WireCell processes RawDigits to Wires in such a way as to remove this offset. Their convention is that all Wires represent the time at which drifting charge would have struck an idealized plane located at the same position as the collection wires.

When applying DetectorProperties' ConvertTicksToX() function on Wires defined in this way, the offsets are removed for a second time, and hits derived from those wires now no longer quite overlap. This plays (subtle) havoc with reconstruction.

Further investigation revealed that ProtoDUNE had independently discovered this problem and was using an adjusted service that did not correct these offsets. This is likely the route we will go in the short term.

Issues for discussion are

1. Are all experiments and software packages using/expecting the same conventions? Is this documented anywhere? We certainly found a corner of DUNE where that wasn't the case, and certainly the convention in use has changed within the last few years.

2. Regardless of the philosophy of what a recob::Wire is supposed to represent, isn't it a big footgun in the interface to have a function that simply takes a tick and wire number and gives the right answer for one of RawDigits and Wires and the wrong answer for the other? With the WireCell conventions, this will be the case one way or another no matter which service implementation we adopt.

Can we adjust the interface in some way (maybe by taking concrete RawDigit and Wire objects? maybe applying a large offset between the two cases so that problems are obvious?) so that it is impossible to confuse the two cases? How about more drastic changes to the interface? I would argue that, since the WireCell-applied correction already introduces a dependency on the drift field, we might as well express Wire objects in terms of a spatial position rather than a time.